Supervisor’s Recommendation

Letter of Approval

Acknowledgement

Our forthright gratefulness goes to Kathmandu Bernhardt College for wholehearted support and for giving us this opportunity to undertake this case study.

We would like to express the deepest gratitude to our supervisor, **Ram Babu Mahato**, for his kind and co-operative support, valuable time and guidance as well as suggestions. We would like to express our gratitude to lecturers from the Department of CSIT at Kathmandu Bernhardt College for providing suggestions for this whole work and cooperative behavior.

We are appreciative of our department's head of Computer Science and Information Technology for providing insightful comments in response to concerns raised about the project.

Last but not least, we would like to express our gratitude to all of our professors, and friends for their unwavering support and inspiration, which enabled us to persevere in our goals and see our project through to completion.

Rojesh Bhattarai (23621/076)

Rakshya Dangol (23617/076)

Sandip Tamang (23625/076)

Abstract

The Construction sites pose significant risks to worker's safety, necessitating the use of Personal Protective Equipment (PPE) for their safety. This project presents a Construction PPE Detection System utilizing computer vision techniques. The system employs YOLO (You Only Look Once) object detection model to identify and classify PPE items such as hardhats, masks, and safety vests in real-time. The graphical user interface (GUI) facilitates both video file analysis and live camera feed processing. Detected objects are outlined with bounding boxes and labeled with corresponding confidence scores. The system enhances safety compliance by providing immediate feedback on PPE usage, thereby reducing the likelihood of accidents and injuries in construction environments. This innovative approach streamlines safety inspections, increases efficiency, and promotes a culture of proactive risk management in construction projects. Moreover, the system's adaptability allows for integration with existing safety protocols and potential expansion to other industries with similar safety concerns.

Keywords: **Construction, Personal Protective Equipment, Object Detection, Computer Vision, YOLO, GUI, Hardhat, Mask, Safety Vest, Proactive Safety, Risk Management.**

Table of Contents

Supervisor's Recommendation ..... **i**

Letter of Approval ..... **ii**

Acknowledgement ..... **iii**

Abstract ..... **iv**

List of Abbreviations ..... **vii**

List of Figures ..... **viii**

List of Tables ..... **ix**

Chapter 1: Introduction **1**

1.1. Introduction 1

1.2. Problem Statement 2

1.3. Objectives 2

1.4. Scope and Limitation 2

1.4.1. Scope 2

1.4.2. Limitations 3

1.5. Development and Methodology 3

1.6. Report Organization 5

Chapter 2: Background Study and Literature Review **7**

2.1. Background Study 7

2.2. Literature Review 8

Chapter 3: System Analysis **10**

3.1. System Analysis 10

3.1.1. Requirement Analysis 10

3.1.2. Feasibility Analysis 11

3.1.3. Analysis 13

Chapter 4: System Design **16**

4.1. Design 16
4.1.1. Interface Design 16
4.2. Algorithm Details 16
Chapter 5: Implementation and Testing 22
5.1. Implementation 22
5.1.1. Tools Used 22
5.1.2. Implementation Details of Modules 22
5.2. Testing 26
5.2.1 Test Cases for Unit Testing 26
5.2.2. Test Cases for System Testing 27
5.3. Result Analysis 28
Chapter 6: Conclusion and Future Recommendations 32
6.1. Conclusion 32
6.2. Future Recommendations 32
References
Appendices

List of Abbreviations

BCE	Binary Cross Entropy
BiFPN	Bidirectional Feature Pyramid Network
CIoU	Complete Intersection over Union
CNN	Convolutional neural network
COCO	Common Object in Context
CoreML	Core Machine Learning
CSP	Cross Stage Partial Network
DL	Deep Learning
FPA	Field-Programmable Gate Arrays
GPS	Global Positioning System
GUI	Graphical User Interface
HOG	Histogram of Oriented Gradients
MAP	Mean Average Precision
NHU	Non-Hardhat Users
NMS	Non-Maximum Suppression
ONNX	Open Neural Network Exchange
PPE	Personal Protective Equipment
ROI	Region of Interest
SDLC	System Development Life Cycle
SSWM	Site Specific Weed Management
SVM	Support Vector Machine
TFLite	TensorFlow Lite
YOLO	You Only Look Once

List of Figures

Figure 1.1: Waterfall Development Life Cycle 4
Figure 1.2: Report Organization 6
Figure 3.1: Use case diagram of Construction PPE Detection System 10
Figure 3.2: GANTT chart of Construction PPE Detection System 13
Figure 3.3: 0 Level DFD of Construction PPE Detection System 14
Figure 3.4: 1 Level DFD of Construction PPE Detection System 15
Figure 4.1: Interface and Dialogue Design 16
Figure 4.2: Visualization of an anchor box in YOLO 16
Figure 4.3: System Flow Diagram of Construction PPE Detection System 17
Figure 4.4: Architecture of YOLOv8 21
Figure 5.1: Recall-Confidence Curve 29
Figure 5.2: Precision-Confidence Curve 29
Figure 5.3: Output Summary 30
Figure 5.4: F1-Confidence Curve 30
Figure 5.5: Precision-Recall Curve 31

List of Tables

Table 5.1: Test Cases for Unit Testing 26
Table 5.2: Test Case 1 for System Testing 27
Table 5.3: Test Case 2 for System Testing 28

Chapter 1: Introduction

1.1. Introduction

A Personal Protective Equipment (PPE) detecting system is a technology designed to identify and monitor the usage of safety gear, such as helmets, and masks by individuals in a given environment. It employs sensors, cameras, or other detection mechanisms to ensure that individuals are wearing the required protective equipment for their safety in specific situations or workplaces.

A range of advanced systems, including the Occupational Safety Assurance System, Compliance Monitoring and Assurance System, and Safety Gear Verification System, leverage sensors, image recognition, and artificial intelligence to enhance safety and compliance. These technologies actively monitor and verify the correct use of personal protective equipment (PPE), ensuring adherence to safety protocols in industrial settings. Tailored for specific regulatory environments, the Compliance Monitoring System fosters a secure workplace, while the Safety Gear Verification System focuses on confirming the usage of safety gear like goggles and masks. Together, these systems exemplify the transformative impact of technology in promoting real-time safety across industries.

The PPE Detection System, incorporating the advanced YOLOV8 algorithm, serves as an intelligent tool for enhancing safety on construction sites. This innovative system adeptly identifies crucial safety equipment such as masks, hardhats, safety vests, gloves, goggles, and shoes worn by workers. It seamlessly operates with both uploaded videos and live camera feeds, enabling real-time safety assessments. Leveraging cutting-edge smart technology, it swiftly verifies compliance with safety protocols, providing an additional layer of vigilance to ensure that everyone is wearing the appropriate safety gear. In essence, the integration of the YOLOV8 algorithm empowers the system to comprehensively detect helmet, goggles, mask, safety vest, gloves, and shoes, contributing to the creation of even safer environments on construction sites.

1.2. Problem Statement

In contemporary industrial and public settings, ensuring the proper utilization of Personal Protective Equipment (PPE) is imperative for safeguarding individuals against potential hazards. However, manual monitoring of PPE compliance is often labor-intensive and prone to errors, leading to gaps in safety protocols. The absence of a streamlined and automated system for real-time PPE detection contributes to an increased risk of accidents and health-related issues.

Existing solutions for PPE detection often lack the desired balance between accuracy, real-time processing, and user accessibility. Current methods may require extensive manual intervention, hindering their effectiveness in dynamic environments. Moreover, the lack of user-friendly interfaces makes these solutions less accessible for non-technical personnel, limiting their widespread adoption.

The "PPE Detection System with YOLO " project seeks to address these challenges by developing a comprehensive solution that integrates the cutting-edge YOLO model. The primary problem to be solved is the need for a reliable and efficient system capable of real-time PPE detection, aiding in the proactive enforcement of safety measures in various industries and public spaces.

1.3. Objectives

The objective is to identify the Personal Protective Equipment (PPE) using “a simple recognition algorithm” from which we can ensure the safety of the people working in different fields. The main objective is given below:

- To build a robust PPE Detection System with GUI interface using yolov8 algorithm.

1.4. Scope and Limitation

1.4.1. Scope

This project aims to develop a Construction PPE Detection System leveraging the YOLO (You Only Look Once) model from the Ultralytics library. The system will facilitate real-time identification of workers' Personal Protective Equipment (PPE) on construction sites, ensuring compliance with safety standards. The application will offer two primary functionalities: video file-based detection and live camera feed detection. Users can select

a video file for retrospective analysis or initiate live monitoring through a user-friendly Tkinter-based GUI. The system will identify and categorize PPE items such as hardhats, masks, safety vests, providing visual feedback on detected objects and their confidence levels. The project addresses critical workplace safety concerns in the construction industry by automating PPE compliance monitoring, thereby contributing to a safer working environment.

1.4.2. Limitations

The project also has some limitation which are highlighted in the following points:

- Environmental conditions, like changing lighting, may affect PPE detection accuracy, leading to potential errors.
- Video quality significantly influences detection accuracy, with low-resolution videos potentially hindering performance.

1.5. Development and Methodology

The Waterfall Model is an advantageous approach in software development projects where requirements are well-defined and unlikely to change significantly throughout the project lifecycle. This model is characterized by its sequential and linear progression through distinct phases, making it particularly suitable for projects with a clear and stable set of requirements from the outset. One key benefit of the Waterfall Model is its simplicity and ease of understanding. The linear nature of the model allows for a systematic and organized development process, making it easier to manage and plan.

The system used Waterfall development methods as the requirements in the project are well-defined and fixed. The system is constructed on 7 phases with each phase of 15 days. Stand up meeting and communication is done by meeting twice in every phase and progress of work was shared through discord.

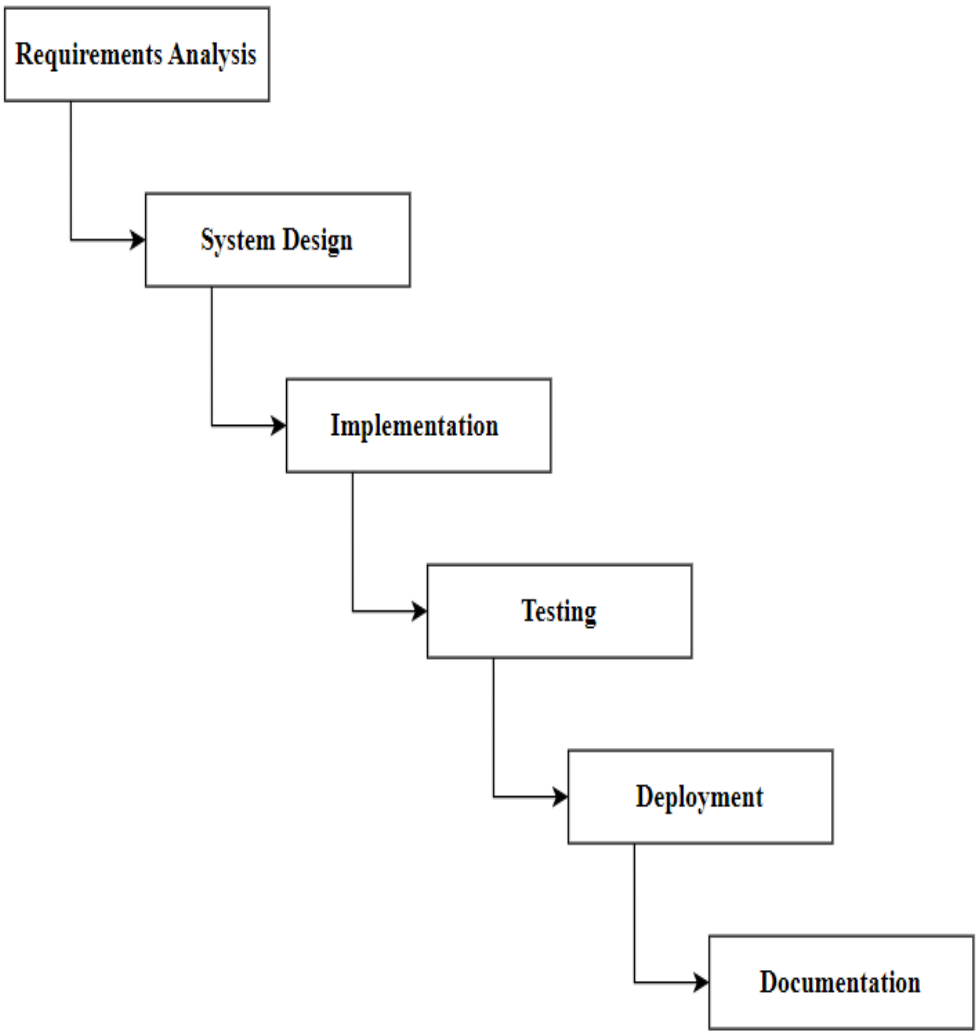


Figure 1.1: Waterfall Development Life Cycle

Work to be done in each phase:

1. Requirements Analysis:

- Identify and document the project requirements, specifying the functionalities and constraints of the PPE Detection System.
- Define the classes of PPE to be detected, the environmental conditions to consider, and the acceptable level of false positives/negatives.

2. System Design:

- Create a comprehensive system design based on the specified requirements, outlining the architecture, components, and interactions.
- Specify the hardware components (e.g., cameras, sensors) and software elements, including the YOLO model, GUI components, and image processing algorithms.

3. Implementation:
    - Begin the implementation phase by coding the functionalities outlined in the system design.
    - Develop the Tkinter-based GUI, integrate the YOLO model for object detection, and implement the video processing logic.
4. Testing:
    - Conduct thorough testing to ensure the proper functionality of each component.
    - Verify the accuracy of PPE detection under various scenarios, test the GUI's responsiveness, and address any issues identified during testing.
5. Deployment:
    - Deploy the Construction PPE Detection System in a controlled environment for initial real-world testing.
    - Gather user feedback and make necessary adjustments based on the system's performance in a practical setting.
6. Documentation:
    - Prepare comprehensive documentation covering the system's architecture, functionalities, and user instructions.
    - Document any known limitations, including environmental conditions that may affect accuracy and potential scenarios where the system might not perform optimally.

1.6. Report Organization

The project is organized into six chapters, each representing a different stage of the project's development. The chapter can be summarized as follows:

Chapter 1: It deals with the introductory part of the project and explains what the project is, how it came into being, the main objectives of the project that are planned to be achieved after the completion, and its scope and limitations.

Chapter 2: This chapter is all about the research process carried out to do the project. It consists of the background idea for the project as well as the study process that is required for this application to be popular among users.

Chapter 3: This chapter deals with the analysis phase. It consists of all the requirements and feasibility analyses needed for the system.

Chapter 4: This chapter is all about the design phase. System design consists of database design, interface design, and process design that is carried out to build this application.

Chapter 5: This chapter is all about the implementation and testing phase. It consists of the different tools used for the implementation of the system along with different test cases for unit testing and system testing. It also consists of the analysis of the results that are obtained.

Chapter 6: This chapter is all about the conclusion and future recommendations phase. The conclusion of the system is written here along with future recommendations of what can be done to make the system more functional.

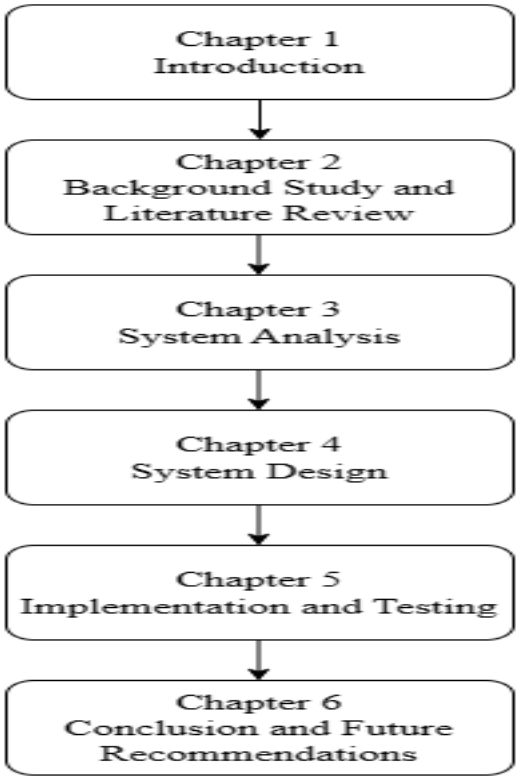


Figure 1.2: Report Organization

Chapter 2: Background Study and Literature Review

2.1. Background Study

YOLOv8 is a family of compound-scaled object detection models trained on the COCO dataset and includes simple functionality for TTA, model assembling, hyperparameter evolution, and export to ONNX, CoreML, and TFLite. YOLO stands for "You Only Look Once" and is an extremely fast object detection framework using a single convolutional network. YOLO is frequently faster than other object detection systems because it looks at the entire image at once as opposed to sweeping it pixel-by- pixel. YOLO does this by breaking an image into a grid, and then each section of the grid is classified and localized (i.e., the objects and structures are established). Then, it predicts where to place bounding boxes [1].

Predicting these bounding boxes is done with regression-based algorithms, as opposed to classification-based ones. Generally, classification-based algorithms are completed in two steps: first, selecting the ROI, then applying the CNN to the regions selected to detect an object(s). YOLOv8 is composed of three convolution layers that predict the location of the bounding boxes (x, y, height, width), the scores, and the object classes. Choosing an activation function is crucial for any deep-learning. YOLOv8 returns three outputs: the classes of the detected objects, their bounding boxes, and the objectness scores. Thus, it uses BCE to compute the class loss and the objectness loss. While CIoU loss to compute the location loss [2].

YOLOv8 utilizes a fully convolutional neural network that can be divided into two main parts: the backbone and the head. The backbone of YOLOv8 is a modified version of the CSPDarknet53 architecture. This architecture consists of 53 convolutional layers and employs a technique called cross-stage partial connections to improve information flow between the different layers of the network. The head of YOLOv8 consists of multiple convolutional layers followed by a series of fully connected layers. These layers are responsible for predicting the bounding boxes, objectness scores, and class probabilities for the objects detected in an image. The Model Backbone is a pre-trained network used to extract rich feature representation for images. This helps reduce the spatial resolution of the image and increase its feature (channel) resolution. The model head is used to perform

the final stage operations. It applies anchor boxes on feature maps and renders the final output: classes, objectness scores, and bounding boxes [3].

One of the key features of YOLOv8 is the use of a self-attention mechanism in the head of the network. This mechanism allows the model to focus on different parts of the image and adjust the importance of different features based on their relevance to the task. Another important feature of YOLOv8 is its ability to perform multi-scale object detection. The model utilizes a feature pyramid network to detect objects of different sizes and scales within an image. This feature pyramid network consists of multiple layers that detect objects at different scales, allowing the model to detect large and small objects within an image [3].

YOLOv8 is an anchor-free model. This means it predicts directly the center of an object instead of the offset from a known anchor box. Anchor free detection reduces the number of box predictions, which speeds up Non-Maximum Suppression (NMS), a complicated post processing step that sifts through candidate detections after inference. Deep learning research tends to focus on model architecture, but the training routine in YOLOv5 and YOLOv8 is an essential part of their success. YOLOv8 augments images during training online. At each epoch, the model sees a slightly different variation of the images it has been provided. One of those augmentations is called mosaic augmentation. This involves stitching four images together, forcing the model to learn objects in new locations, in partial occlusion, and against different surrounding pixels. This augmentation is empirically shown to degrade performance if performed through the whole training routine. It is advantageous to turn it off for the last ten training epochs [2] [3].

YOLOv8 is highly scalable and can be used for small-scale as well as large-scale projects. YOLOv8 can run on a range of hardware, from desktop CPUs to GPUs, dedicated hardware like FPGAs and YOLOv8 is open source, which means developers can access the source code and modify it as needed.

2.2. Literature Review

In recent years several works have been done for the safety of the workers in construction sites based on CV and deep learning (DL). DL methods have a strong ability to self-learning from useful features. Region-based CNNs (R-CNNs) were used to identify whether a worker wears a hardhat or not on the construction site. Their precision and recall rate were

approximately 95.7% and 94.9% for the non-hardhat users (NHU). The authors did not consider hardhat detection rather they only detect the non-hardhat users [4]. Histograms of oriented gradient (HOG) was used by to extract head features from images. Feeding the extracted features into the support vector machine (SVM) the authors try to classify whether one worker wears a helmet or not [5].

The YOLOv3 architecture was used to predict four classes such as NOT SAFE, SAFE, No Hardhat, and No Jacket. The authors trained their model using 2,509 images that were collected from video recordings from the construction sites and internet-based collections. The average precision and recall rate are 96% on the test data. Authors try to make the CV more trustable in the real world by doing an alarm system by integrating it and also a reporting system with a certain time period [6]. Human identity recognition and helmet detection were build using YOLOv3 in a construction site [7]. In 2020 the PPE detection was made using YOLOv3 algorithm. The authors proposed a dataset named Pictor-v3, which contains 1,500 images and corresponding annotations. They reported the highest performance is 72.3% mean average precision (mAP) with 11 FPS. The AP for vest and helmet is 84.96% and 79.81% respectively [8]. An improved weighted bi-directional feature pyramid network (BiFPN) for hardhat wearing detection, they also try to detect the color of hardhat into the construction site. The authors showed 87.04% mAP yields by their proposed method for five classes [9]. A dataset named CHV and use YOLO family architecture (YOLOv3, YOLOv4, and YOLOv5) to detect the PPE of the workers. The authors found that YOLOv5x outperforms the others method and the mAP is 86.55% for six classes. They proclaimed that 52 FPS for one single image is processed by the YOLOv5s model. This study is relevant and complementary to the aforementioned research. An attempt has been made to further enhancement of the above studies through this study. A new anchor-free training architecture is proposed for PPE detection into the construction site. This study also tries to explore the latest dataset for PPE detection both increasing data size and the number of classes [10].

Chapter 3: System Analysis

3.1. System Analysis

3.1.1. Requirement Analysis

1. Functional Requirements

The system used Waterfall development methods hence the requirements in this presented project are well-defined and fixed. Initially brainstorming was done to find out what the possible requirements were. And the results obtained were: a high-definition camera, a system in which the dataset is to be stored and a project is implemented, a field agent whose main task is to take detailed picture of the PPE kits, and a system for testing its accuracy and goal of the project. Later on, a feasibility study was done to verify whether the project could be actualized or not.

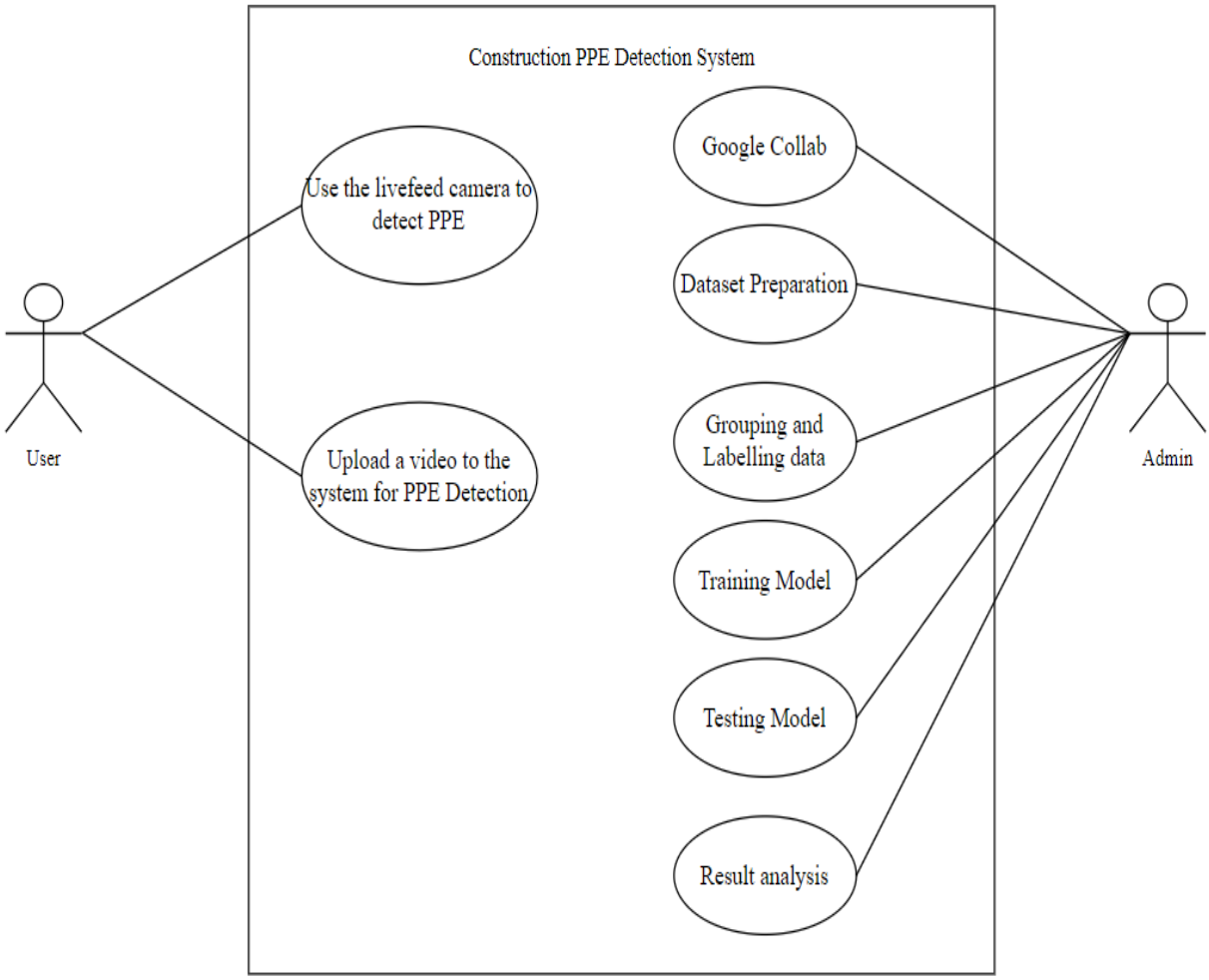


Figure 3.1: Use case diagram of Construction PPE Detection System

1. Admin function

The admin's role also includes monitoring the system's performance, identifying and resolving any issues that may arise, and implementing any necessary updates or improvements to ensure the system runs smoothly.

2. User function

Users will Capture videos of Worker under the permission of companies, and distribute those images. They also use our system to verify worker wearing PPE kits.

2. Non-Functional Requirements

a. User Friendly

The term user-friendly is self-explanatory. Users can easily use the PPE detection system. The software uses a simplified design and navigation, as well as simple language on the content to improve user-friendliness.

b. Easy Access

It can be used anywhere without an Internet Connection.

c. Information Accuracy

The system makes sure to avoid making mistakes during data and information retrieval. Thus, the system provides the accuracy of information.

d. Speed of Application

The speed of the application depends on the System configuration and. Systems with good configuration will most definitely give a lag free experience.

3.1.2. Feasibility Analysis

1. Technical

The technical aspects of the Construction PPE Detection System include the utilization of High-Definition Cameras to capture images of individuals wearing PPE from various angles. Python is selected as the programming language, providing a versatile and widely used platform for development. Google Drive serves as the storage solution for the dataset, ensuring convenient and accessible data management. The system requires PyCharm for installation and dataset compilation, and it doesn't demand a high-performance device, making it accessible on standard computers. While the procurement of High-Definition cameras may present a cost consideration, proper budget allocation can easily address this

concern. Consequently, the proposed project is deemed technically feasible, leveraging practical and accessible tools for effective PPE detection.

2. Operational

The development and operation of the system will be carried out by our team members, with oversight and operation managed by a trained technician to be hired for this purpose. Diverse datasets will be employed for effective PPE identification in individuals.

Training for personnel to adeptly handle the system will be facilitated by the system developer, ensuring a smooth transition into system operation. Any challenges arising from system complexity will be addressed through consistent debugging and rigorous testing procedures, ensuring the reliability and effectiveness of the PPE detection system.

3. Economic

The primary expense associated with this project is directly linked to the frequency of field visits required for dataset creation or updates, coupled with the cost of purchasing the necessary cameras for capturing sample data. An organized budget can be allocated to cover all cost factors, encompassing camera procurement, transportation expenses for field visits, system maintenance, software installation, and other related expenditures.

The project's minimal budget needs make it easily manageable for the team, streamlining resource allocation and reducing financial strain. This financial feasibility allows a greater focus on development and implementation, promoting flexibility for adapting to evolving requirements. Overall, the manageable budget enhances the project's viability and potential for successful implementation.

4. Schedule

The project's anticipated schedule is initially mapped out using a Gantt chart, providing a visual representation that allows the team to compare projected completion dates with the real-time project schedule. This comparison aids in assessing the project's progress and identifying any deviations from the planned timeline. Effective management by the team captain, coupled with the enthusiasm of team members, is integral to ensuring that the project remains on track and is completed within the set deadline. The Gantt chart features a list of activities on the left, aligned with a suitable time scale along the top, offering a

comprehensive overview of task durations and dependencies for seamless project management.

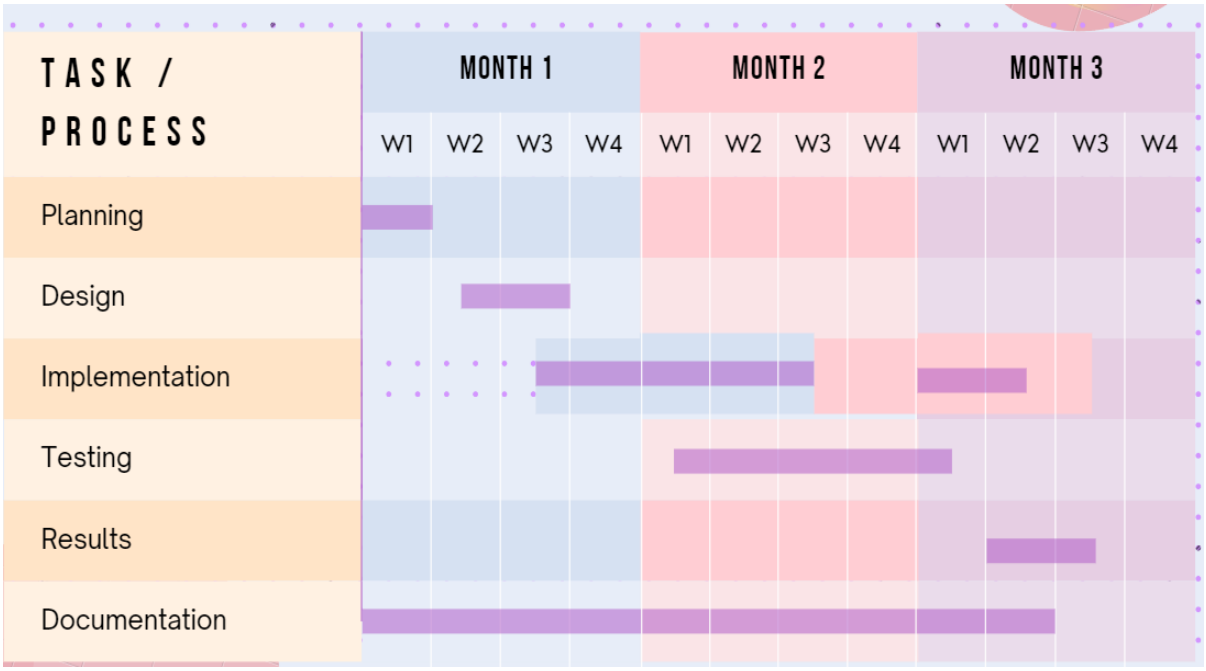


Figure 3.2: GANTT chart of Construction PPE Detection System

3.1.3. Analysis

In the analysis phase of our project, we opted for a methodical and structured approach to 8application development, prioritizing systematic and organized methodologies. The rationale behind selecting this approach was rooted in the necessity for clear, maintainable, and scalable development practices throughout the project's lifecycle. Adhering to a structured methodology aimed to improve the overall efficiency of the development process, encourage collaborative efforts among team members, and support the modularity of the codebase. This approach not only simplifies the coding process but also ensures that the application remains comprehensible and adaptable for future enhancements or modifications. By breaking down complex tasks into manageable components, the structured approach facilitated a logical and systematic progression from design to implementation. Ultimately, our decision to follow a structured approach reflects our commitment to constructing a resilient and easily maintainable application in accordance with established software development best practices.

- Data Flow Diagram (DFD)

A Data Flow Diagram is a graphical representation that illustrates how data is processed and transferred within a system. It is a modeling technique used in the field of software engineering and systems analysis to visualize the flow of data between different components or processes within a system.

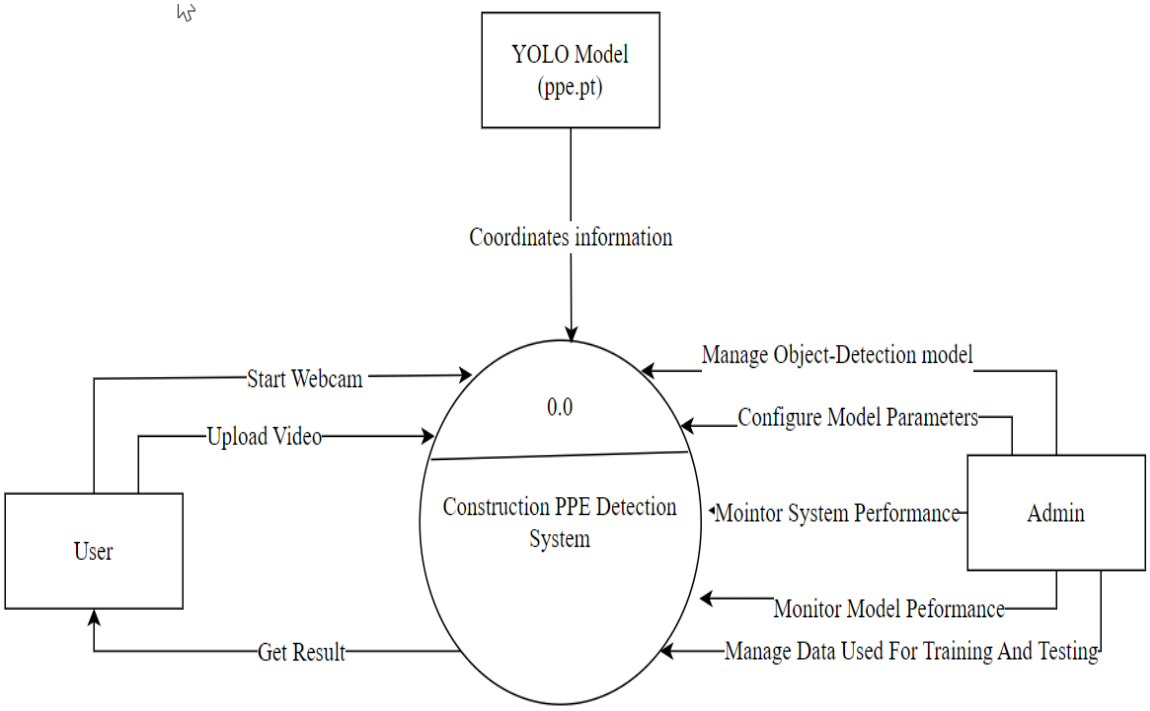


Figure 3.3: 0 Level DFD of Construction PPE Detection System

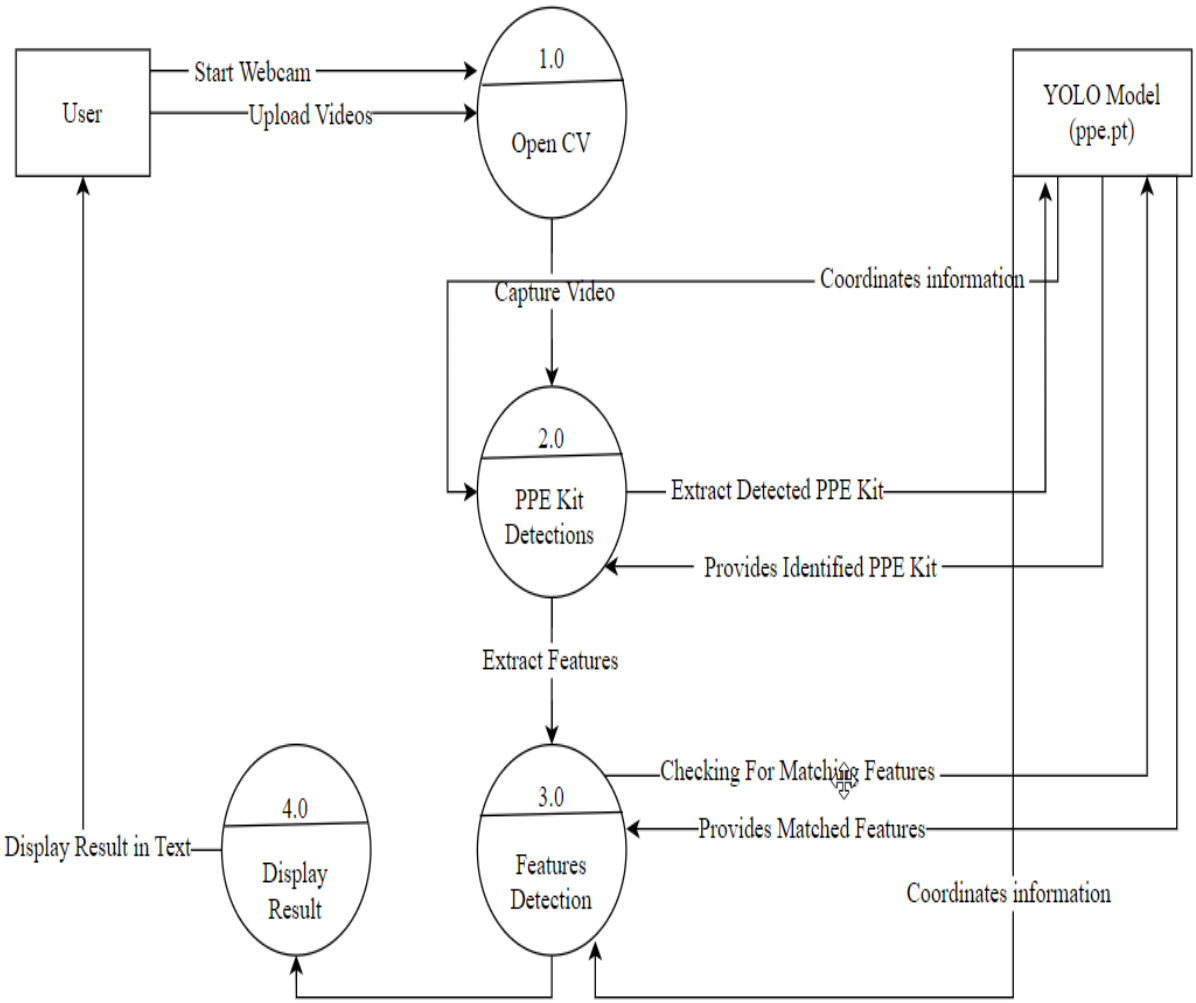


Figure 3.4: 1 Level DFD of Construction PPE Detection System

Chapter 4: System Design

4.1. Design

System design is the process of defining a system's architecture, components, interface, and data to satisfy preset requirements. It is a critical phase in a system's life cycle. During this stage, the ensuing design and diagrams were produced, and they have through a thorough analysis.

4.1.1. Interface Design

Interface and dialogue design are crucial components in the development of user-friendly software, websites, applications, and other digital products. They involve creating visually appealing and intuitive interfaces, as well as designing clear and effective communication between the user and the system.

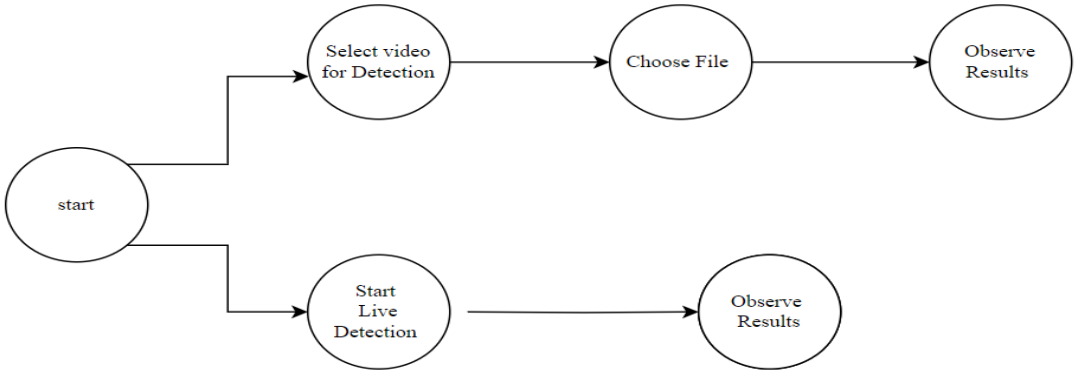


Figure 4.1: Interface and Dialogue Design

4.2. Algorithm Details

YOLO (You Only Look Once) is a popular object identification approach that predicts bounding boxes and class probabilities for each object in a picture using a single neural

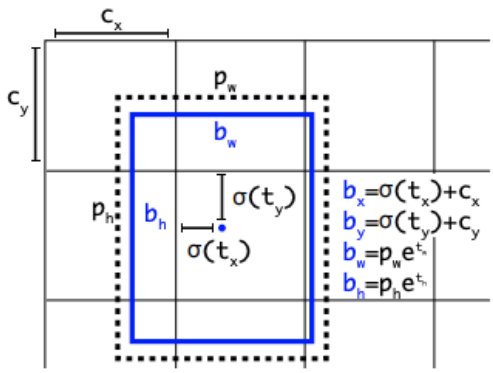


Figure 4.2: Visualization of an anchor box in YOLO

network. YOLOv8 is the most recent version of the YOLO algorithm, and it improves on earlier versions by integrating new modules such as spatial attention, feature fusion, and context aggregation. These enhancements result in faster and more accurate object detection, elevating YOLOv8 to the forefront of object detection algorithms.

The equation for visualization of an anchor box in YOLO are as follows:

$$b_x = \sigma(t_x) + c_x \quad (1)$$

$$b_y = \sigma(t_x) + c_x \quad (2)$$

$$b_w = p_w e^{tw} \quad (3)$$

$$b_h = p_h e^{th} \quad (4)$$

where:

t_x, t_y are the predicted offsets.

t_w, t_h are the predicted scale parameters.

p_w, p_h are the width and height of the bounding box prior

cx, cy are the coordinates of the upper left corner of the grid cell

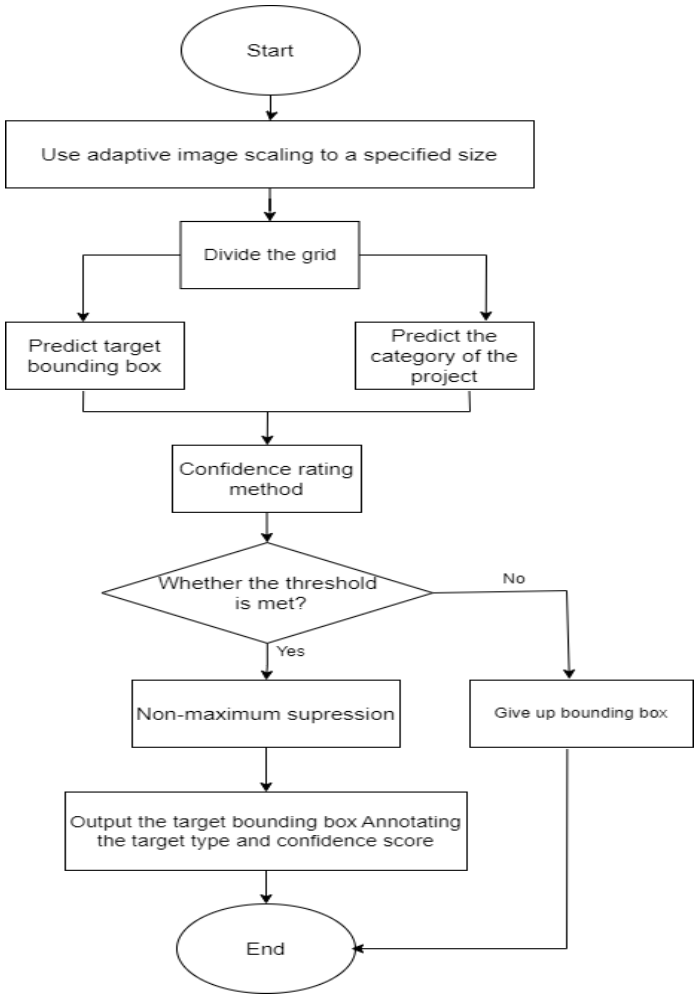


Figure 4.3: System Flow Diagram of Construction PPE Detection System

The key operations in the YOLOv8 algorithm are:

1. **Convolutional Neural Network**

   Yolov8 utilizes a deep convolutional neural network to extract features from the input image. The network consists of multiple convolutional layers with different filter sizes and stride values. The convolutional layers help to capture the spatial and semantic information of the input image.

2. **Feature Extraction**

   The output feature map from the convolutional neural network is passed through additional convolutional and pooling layers to down-sample it. The feature extraction process aims to create a feature map that can efficiently represent objects at different scales and locations within the image.

3. **Bounding Box Prediction**

   For each grid cell in the feature map, Yolov8 predicts multiple bounding boxes using anchor boxes. The anchor boxes are pre-defined bounding boxes of different scales and aspect ratios. Yolov8 uses the anchor boxes to predict the coordinates, width, and height of each bounding box relative to the cell's spatial location.

4. **Objectness Score Prediction**

   Yolov8 predicts the objectness score for each bounding box to indicate whether an object is present within the bounding box or not. The objectness score is represented by a confidence value that ranges from 0 to 1.

5. **Class Prediction**

   Yolov8 predicts the class probabilities for each bounding box based on a fixed number of predefined classes. The class probabilities are calculated using a SoftMax function over the classes.

6. **Non-max Suppression**

   Yolov8 applies a non-max suppression algorithm to remove duplicate or overlapping bounding boxes. The non-max suppression algorithm filters out bounding boxes with low objectness scores or high overlap with other bounding boxes.

7. **Loss Function**

   Yolov8 uses a combination of localization loss, objectness loss, and class loss to train the network. The localization loss measures the difference between the predicted

bounding boxes and the ground truth bounding boxes. The objectness loss calculates the difference between the predicted objectness scores and the ground truth objectness scores. The class loss evaluates the difference between the predicted class probabilities and the ground truth class labels.

Localization Loss = λ_coord * (Loss_x + Loss_y + Loss_w + Loss_h)……..…..(5)
where:
Loss_x: Measures the discrepancy between the predicted x-coordinate of the bounding box and the ground truth x-coordinate.
Loss_y: Measures the discrepancy between the predicted y-coordinate of the bounding box and the ground truth y-coordinate.
Loss_w: Measures the discrepancy between the predicted width of the bounding box and the ground truth width.
Loss_h: Measures the discrepancy between the predicted height of the bounding box and the ground truth height.
λ_coord: A hyperparameter that balances the influence of the localization loss compared to other components of the overall loss function, such as the confidence loss.

Class Loss = λ_class * (Loss_class1 + Loss_class2 + ... + Loss_classN)……...(6)
where:
Loss_class1, Loss_class2, ..., Loss_classN: Measure the discrepancy between the predicted class probabilities and the ground truth class labels for each object.
λ_class: A hyperparameter that balances the influence of the class loss compared to other components of the overall loss function, such as the localization loss.

Objectness Loss = λ_obj * (Loss_obj)……(7)
where:
Loss_obj: Measures the discrepancy between the predicted objectness score and the ground truth objectness label for each grid cell.
λ_obj: A hyperparameter that balances the influence of the objectness loss compared to other components of the overall loss function, such as the localization loss and class loss.

YOLOv8 provides a unified framework for training models for performing object detection, instance segmentation, and image classification. The users can use a single model for all three tasks, simplifying the training process.

The Yolov8 algorithm is built on the principles of YOLO-based object detection and includes several modifications and enhancements to improve the accuracy and speed of object detection.

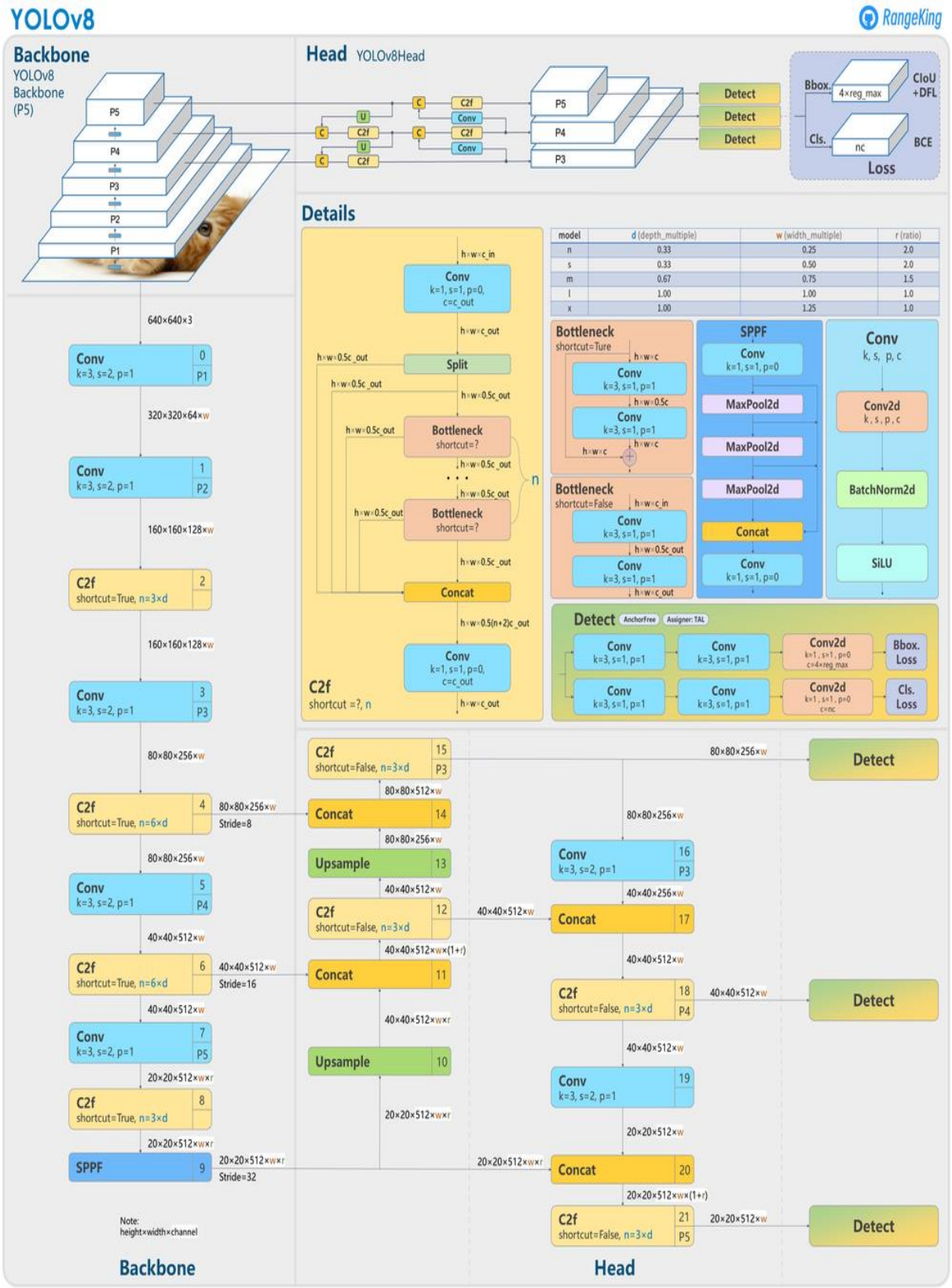


Figure 4.4: Architecture of YOLOv8

Chapter 5: Implementation and Testing

5.1. Implementation

The system components as identified in the design specification from the design phase and the software requirements specification from the analysis phase have been built. This section documents the tools and testing used for this project.

5.1.1. Tools Used

a. Frontend tools

1. Draw.io

   It is a website that provides online diagramming tools. This tool was used to create various diagrams such as use case diagrams, DFD and context diagrams, and system flow diagrams which helps in understanding the overall flow of the application.

b. Backend tools

1. Google Collab

   Collab allows anybody to write and execute arbitrary Python code through the browser and is especially well suited to machine learning, data analysis, and education. More technically, Collab is a hosted Jupyter notebook service that provides access free of charge to computing resources including GPUs.

2. Pytorch

   PyTorch is an optimized Deep Learning tensor library based on Python and Torch and is mainly used for applications using GPUs and CPUs. PyTorch is favored as it uses dynamic computation graphs and is completely Pythonic.

3. Pycharm

   PyCharm is an integrated development environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django.

5.1.2. Implementation Details of Modules

Our System is mainly divided into 3 modules. They are as follows:

1. Train Module

Here, Algorithm YOLOv8 is loaded. The images are preprocessed by scaling to 512 *512 pixels. Inference of model is drawn. The model is trained for 100 epochs. The inference is done to find best fitness result saved in best.pt.

Pseudocode for training the dataset

```
#Load the YOLOv8 model

model = YOLO('yolov8.pt')


# Load and preprocess the input image
Preprocesses a batch of images by scaling and converting to float.
def preprocess_batch(self, batch)
        batch['img'] = batch['img'].to(self.device, non_blocking=True).float() / 255
        return batch


 # Perform inference using the YOLOv8 model
def __call__(self, epoch, fitness):
        """
        Check whether to stop training

        Args:
        epoch (int): Current epoch of training
        fitness (float): Fitness value of current epoch

        Returns:
        (bool): True if training should stop, False otherwise
        """
        if fitness is None:  # check if fitness=None (happens when val=False)
                return False

        if fitness >= self.best_fitness:  # >= 0 to allow for early zero-fitness stage of training
                self.best_epoch = epoch
                self.best_fitness = fitness
```

```
            delta = epoch - self.best_epoch  # epochs without improvement
            self.possible_stop = delta >= (self.patience - 1)  # possible stop may occur
            next epoch
            stop = delta >= self.patience  # stop training if patience exceeded
        if stop:
            LOGGER.info(f'Stopping training early as no improvement observed in last
            {self.patience} epochs. '
            f'Best results observed at epoch {self.best_epoch}, best model saved as
            best.pt.\n'
            f'To update EarlyStopping(patience={self.patience}) pass a new patience
            value, '
            f'i.e. `patience=300` or use `patience=0` to disable EarlyStopping.')
            return stop
```

#For training the model

To train the model, a train module for 90 epochs was used. Data showing the directory path of a dataset.

```
! yolo task=detect mode=train model=yolov8l.pt data= data.yaml epochs=90 imgsz=512
plots=True
```

2. Validation Module

The best.pt and last.pt is obtained from the Train module. Best.pt represents the best set of weights found during training and determined based on the performance of the model on the validation set.

Last.pt represents the most recent set of weights that were saved during training. Using these pt, bounding boxes, class labels, and confidence scores are extracted. Non-maximum suppression was applied to remove overlapping boxes. IoU is way to measure how much two areas overlap. IoU is used as a threshold parameter for NMS.

Pseudo code for validating module

```
# Extract the bounding boxes
def box_area(box):
    return (box[2] - box[0]) * (box[3] - box[1])
#Extract class labels
def process_cls_preds(self, preds, targets):
```

```
        preds, targets = torch.cat(preds)[:, 0], torch.cat(targets)
        for p, t in zip(preds.cpu().numpy(), targets.cpu().numpy()):
                self.matrix[t][p] += 1
#calculating IOU
# Intersection area
inter = (b1_x2.minimum(b2_x2) - b1_x1.maximum(b2_x1)).clamp(0) * \
(b1_y2.minimum(b2_y2) - b1_y1.maximum(b2_y1)).clamp(0)

# Union Area
union = w1 * h1 + w2 * h2 - inter + eps

# IoU
iou = inter / union
# Apply non-maximum suppression to remove overlapping boxes
c = x[:, 5:6] * (0 if agnostic else max_wh)  # classes
boxes, scores = x[:, :4] + c, x[:, 4]  # boxes (offset by class), scores
i = torchvision.ops.nms(boxes, scores, iou_thres)  # NMS
i = i[:max_det]  # limit detections
if merge and (1 < n < 3E3):  # Merge NMS (boxes merged using weighted mean)
        # Update boxes as boxes(i,4) = weights(i,n) * boxes(n,4)
        iou = box_iou(boxes[i], boxes) > iou_thres  # iou matrix
        weights = iou * scores[None]  # box weights
        x[i, :4] = torch.mm(weights, x[:, :4]).float() / weights.sum(1, keepdim=True)  #
        merged boxes
if redundant:
        i = i[iou.sum(1) > 1]  # require redundancy


output[xi] = x[i]
if mps:
        output[xi] = output[xi].to(device)
        if (time.time() - t) > time_limit:
        LOGGER.warning(f'WARNING NMS time limit {time_limit:.3f}s exceeded')
        break # time limit exceeded
return output
```

3. Test Module

The videos were used to detect the (hardhat, mask and safety-vest) in those videos. The test module used the trained and validate module and display the final detected objects.

#For Testing Videos using test module

Detect.py is a test module that uses the weight best.pt (output of train module). A video of 10 sec containing a person wearing hardhat and a safety-vest is selected as output. The detection was then studied keeping an eye on the confidence score.

5.2. Testing

Testing is the final process in the SDLC. Here the developed system is tested to find out errors and fix them. It is one of the most important procedures of the system development life cycle. To verify and validate the system a few tests were performed.

5.2.1 Test Cases for Unit Testing

The smallest testable parts of an application, called units, are individually and independently inspected for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff. The main objective of unit testing is to isolate written code to test and determine if it works as intended. Following test cases were performed for unit testing.

Table 5.1: Test Cases for Unit Testing

S. N.	Test Cases	Input	Expected Outcome	Actual Outcome	Status
1	Check with video containing (Hardhat, Mask, Safety-Vest).	Video of .mp4 format.	Detection of (Hardhat, Mask, Safety-Vest).	(Hardhat, Mask, Safety-Vest) using bounding boxes.	Pass
2	Check with video containing no (Hardhat, Mask, Safety-Vest).	Video of .mp4 format.	No (Hardhat, Mask, Safety-Vest) Detected.	No (Hardhat, Mask, Safety-Vest) Detected.	Pass
3	Choose live Detection.	Live feed from web	Live camera is active and live	Live camera is active and live	Pass

		camera of laptop.	video is displayed in the interface.	video is displayed in the interface.	
4	Choose video for Detection.	Browse video file.	Video is chosen.	Video is chosen and displayed in interface.	Pass

5.2.2. Test Cases for System Testing

System testing, also known as system-level testing or system integration testing, is the process by which a quality assurance (QA) team assesses how the many components of an application interact with one another in the overall, integrated system or application. System testing ensures that an application executes its functions as intended. It is a sort of black-box testing that focuses on application functionality. Following test cases were performed for system testing.

Table 5.2: Test Case 1 for System Testing

Steps	Test Steps	Inputs	Expected Outcome	Actual Outcome	Results
1	Navigate to PyCharm and run the main.py.		The laptop's integrated webcam needs to activate, displaying live video on the interface. The system should identify and mark (Hardhat, Mask, Safety-Vest, No Hardhat, No Mask, No Safety-Vest) by outlining them with bounding boxes.	The laptop's integrated webcam is activated, displaying live video on the interface. The system identifies and marks (Hardhat, Mask, Safety-Vest, No Hardhat, No Mask, No Safety-Vest) by outlining them with bounding boxes.	Pass
2	Choose start live detection.	Live Web camera.			

Table 5.3: Test Case 2 for System Testing

Steps	Test Steps	Input	Expected Outcome	Actual Outcome	Result
1	Navigate to PyCharm and run the main.py.		The system should open and display the chosen video within the interface. It should then proceed to detect and label items such as (Hardhat, Mask, Safety-Vest, No Hardhat, No Mask, No Safety-Vest) by outlining them with bounding boxes.	The system opened and displayed the video within the interface. It then detected and labeled items such as (Hardhat, Mask, Safety-Vest, No Hardhat, No Mask, No Safety-Vest) by outlining them with bounding boxes.	Pass
2	Choose select video for detection				
3	Upload the media in which (Hardhat, Mask, Safety-Vest) is to be detected	ppe-1.mp4			

5.3. Result Analysis

The results of different inputs were done in the unit testing. The major analysis of the testing was the detection of the (Hardhat, Mask, Safety-Vest) with the help of the Construction PPE detection system. The system was tested on the devices of our project team members. A few data from the dataset were tested and the expected results were obtained. A few testing cases failed but that can be recovered in the future. In this project, the detection of the (Hardhat, Mask, Safety-Vest) was tested. The system was trained from the dataset taken from Roboflow. Further improvements to the system can be done by training the model sufficiently using a plethora of data avoiding the cases of underfitting and overfitting.

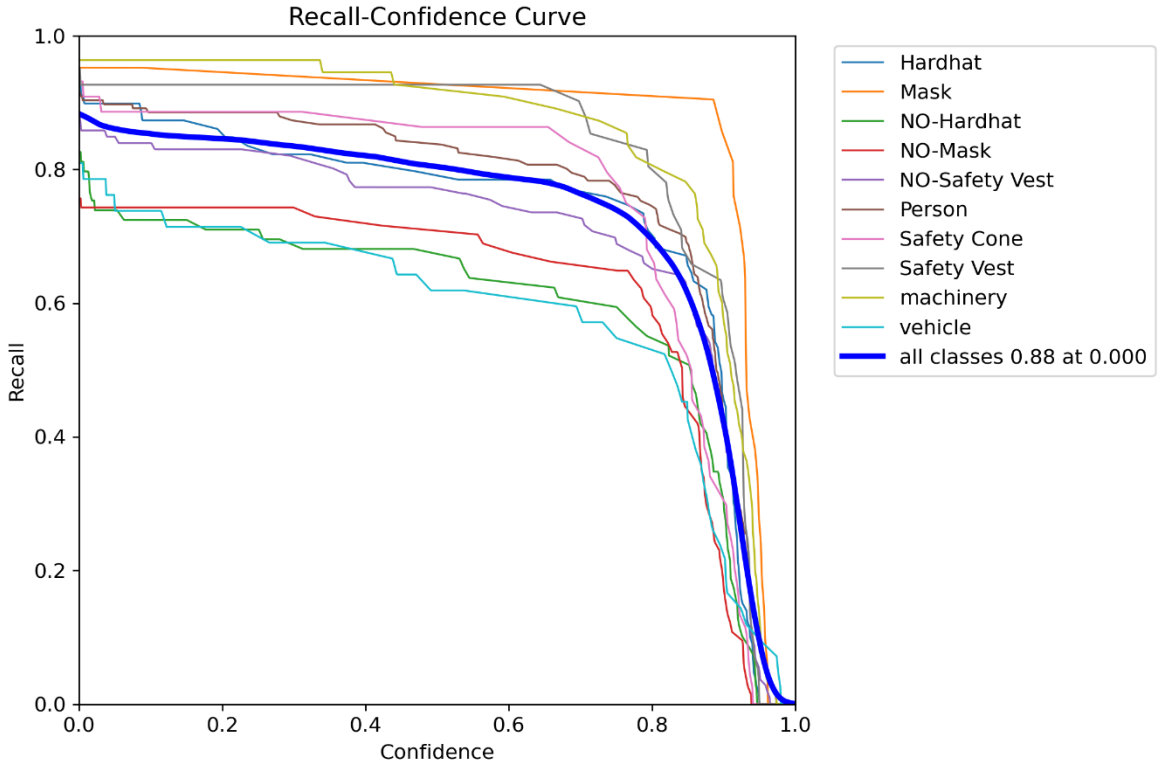


Figure 5.1: Recall-Confidence Curve

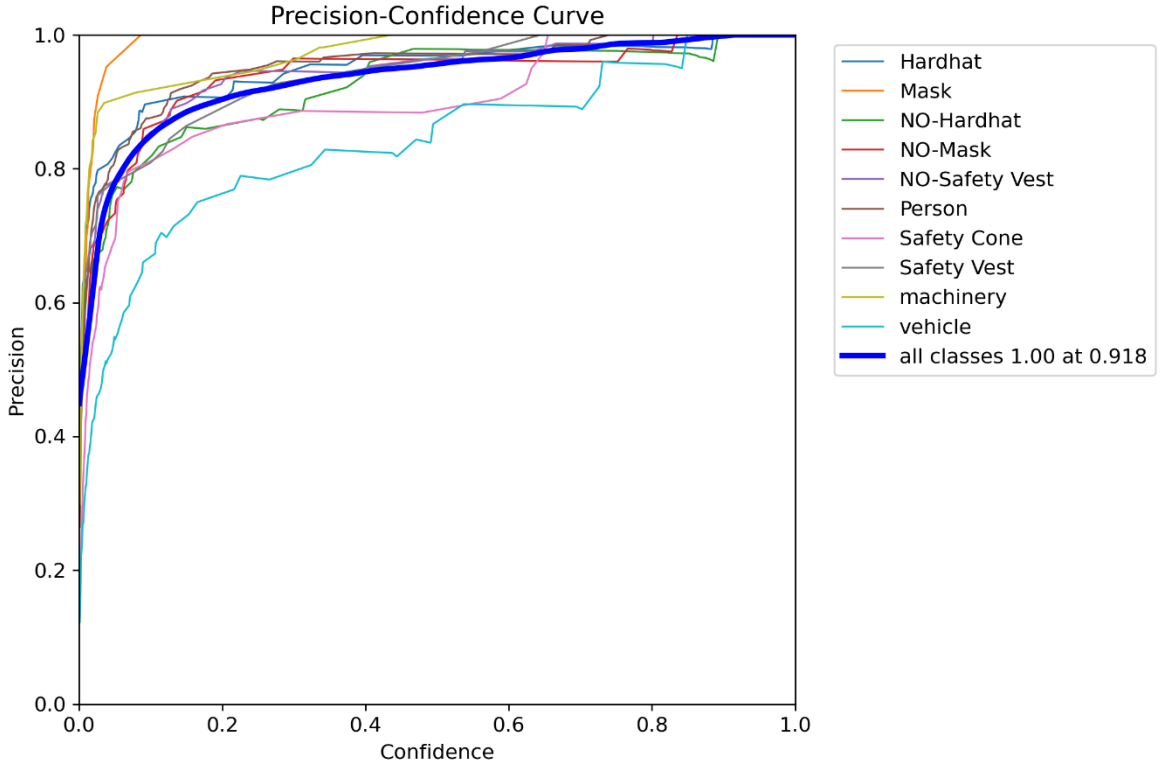


Figure 5.2: Precision-Confidence Curve

```
Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.221 🚀 Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 268 layers, 43614318 parameters, 0 gradients, 164.9 GFLOPs
```

Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 4/4 [00:06<00:00, 1.54s/it]
all	114	697	0.933	0.832	0.884	0.649
Hardhat	114	79	0.956	0.822	0.919	0.68
Mask	114	21	1	0.938	0.963	0.754
NO-Hardhat	114	69	0.906	0.681	0.815	0.507
NO-Mask	114	74	0.964	0.732	0.785	0.547
NO-Safety Vest	114	106	0.945	0.811	0.877	0.62
Person	114	166	0.96	0.869	0.913	0.667
Safety Cone	114	44	0.886	0.885	0.91	0.616
Safety Vest	114	41	0.936	0.927	0.944	0.732
machinery	114	55	0.975	0.964	0.97	0.805
vehicle	114	42	0.806	0.69	0.747	0.557

```
Speed: 0.3ms preprocess, 19.2ms inference, 0.0ms loss, 3.4ms postprocess per image
Results saved to runs/detect/train
```

Figure 5.3: Output Summary

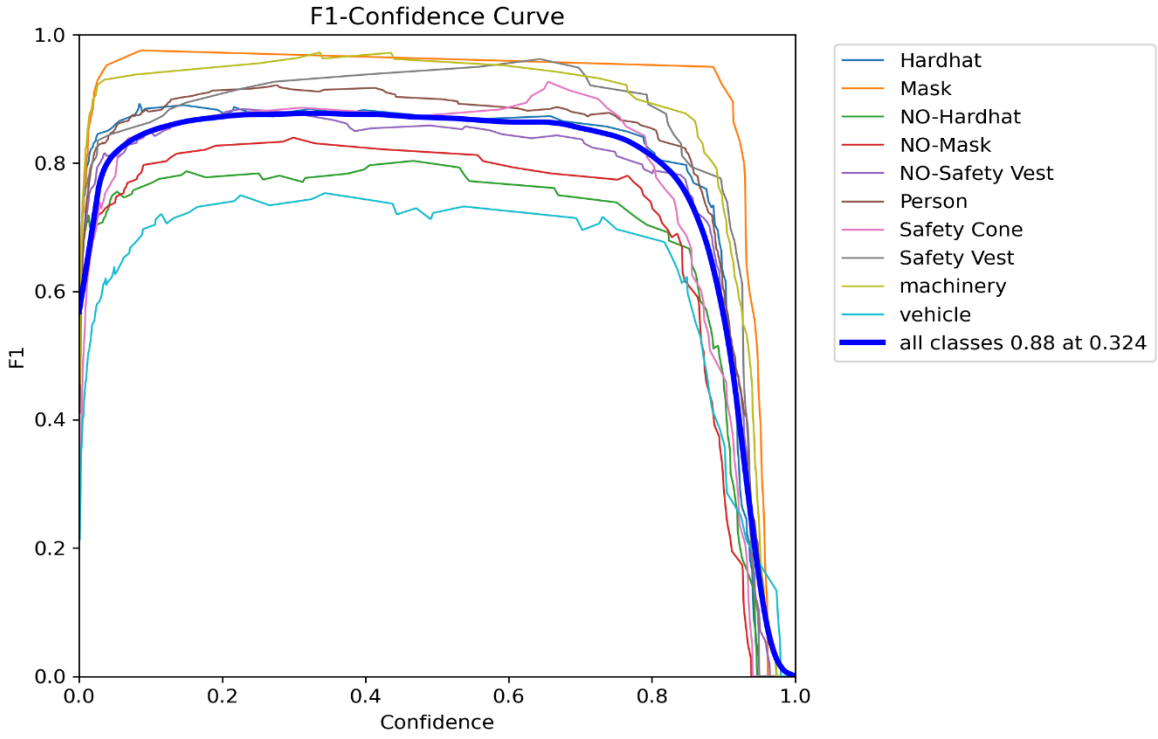


Figure 5.4: F1-Confidence Curve

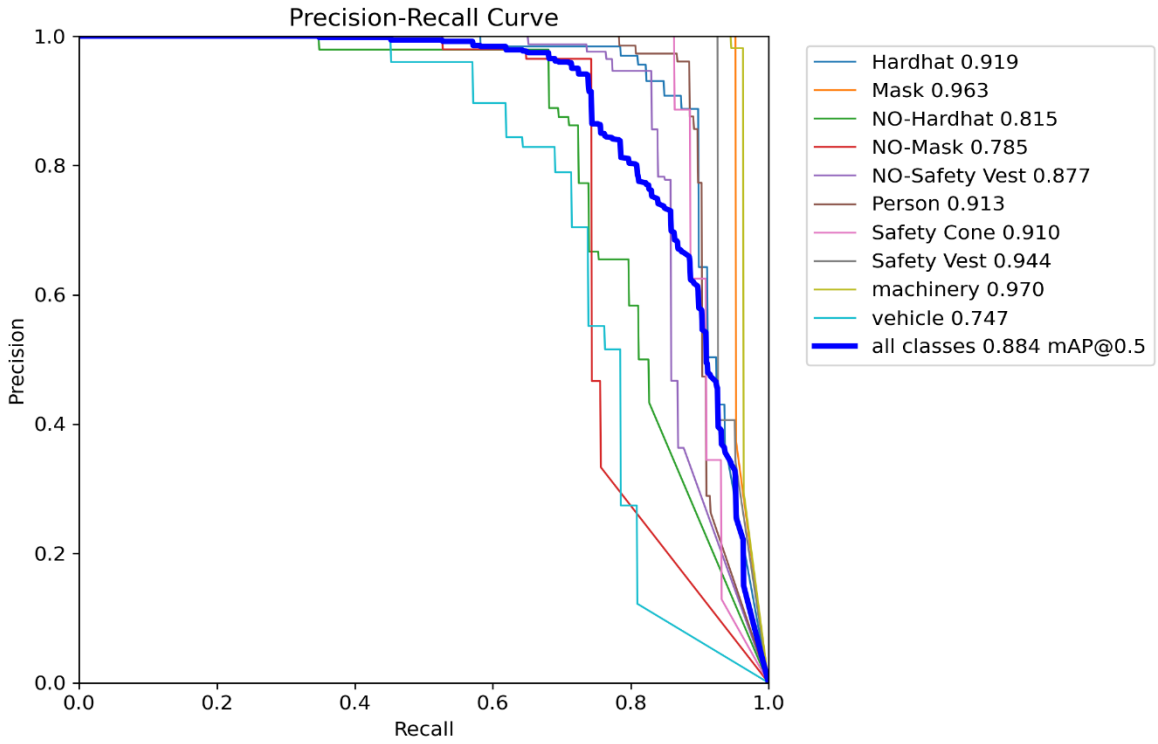


Figure 5.5: Precision-Recall Curve

Chapter 6: Conclusion and Future Recommendations

6.1. Conclusion

The system of Construction PPE detection was developed to demonstrate the significance of AI in the construction sector. The system detects (Hardhat, Mask, Safety-Vest) using the YOLOv8 algorithm. Certain tests were performed in the system to obtain the accuracy of the system and the precision was noted to be 88.4% at 50% threshold.

6.2. Future Recommendations

Future plans include adding more features to the system, such as improving its ability to handle a larger dataset. The detecting system will also undergo enhancements to make it more user-friendly. Moreover, the system is designed to be cost-effective, allowing it to be used by any construction company, without the need for expensive equipment to identify (Hardhat, Mask, Safety-Vest). Finally, a bilingual front end will facilitate communication with users of all racial and ethnic origins.

Reference

[1]D. Davies, "Weights and Biases," 24 December 2022. [Online]. Available: https://wandb.ai/onlineinference/YOLO/reports/YOLOv5-Object-Detection-on- Windows-Step-By-Step-Tutorial---VmlldzoxMDQwNzk4?fbclid=IwAR10E-_pXSIoeiYMxrQu9l4sBz4S7z2BeE_SrF18PdCgnAbEN2o26ndROcg.

[2]F. Jacob Solawetz, "roboflow," [Online]. Available: https://blog.roboflow.com/whats-new-in-yolov8/.

[3]A. Mehra, "LABELLERR," 16 April 2023. [Online]. Available: https://www.labellerr.com/blog/understanding-yolov8-architecture-applications-features/.

[4] Fang et al. (2018) Fang Q, Li H, Luo X, Ding L, Luo H, Rose TM, An W. Detecting non-hardhat-use by a deep learning method from far-field surveillance videos. Automation in Construction. 2018;85:1–9. doi: 10.1016/j.autcon.2017.09.018.

[5] Zhu, Park & Elsafty (2015) Zhu Z, Park M-W, Elsafty N. Automated monitoring of hardhats wearing for onsite safety enhancement. 5th International/11th construction specialty conference, Vancouver, Canada, 8-10 June 2015, 138.2015.

[6] Delhi, Sankarlal & Thomas (2020) Delhi VSK, Sankarlal R, Thomas A. Detection of Personal Protective Equipment (PPE) compliance on construction site using Computer Vision based Deep Learning techniques. Frontiers in Built Environment. 2020;6:136. doi: 10.3389/fbuil.2020.00136.

[7] Wang et al. (2021a) Wang J, Zhu G, Wu S, Luo C. Workers helmet recognition and identity recognition based on deep learning. Open Journal of Modelling and Simulation. 2021a;9(2):135–145.

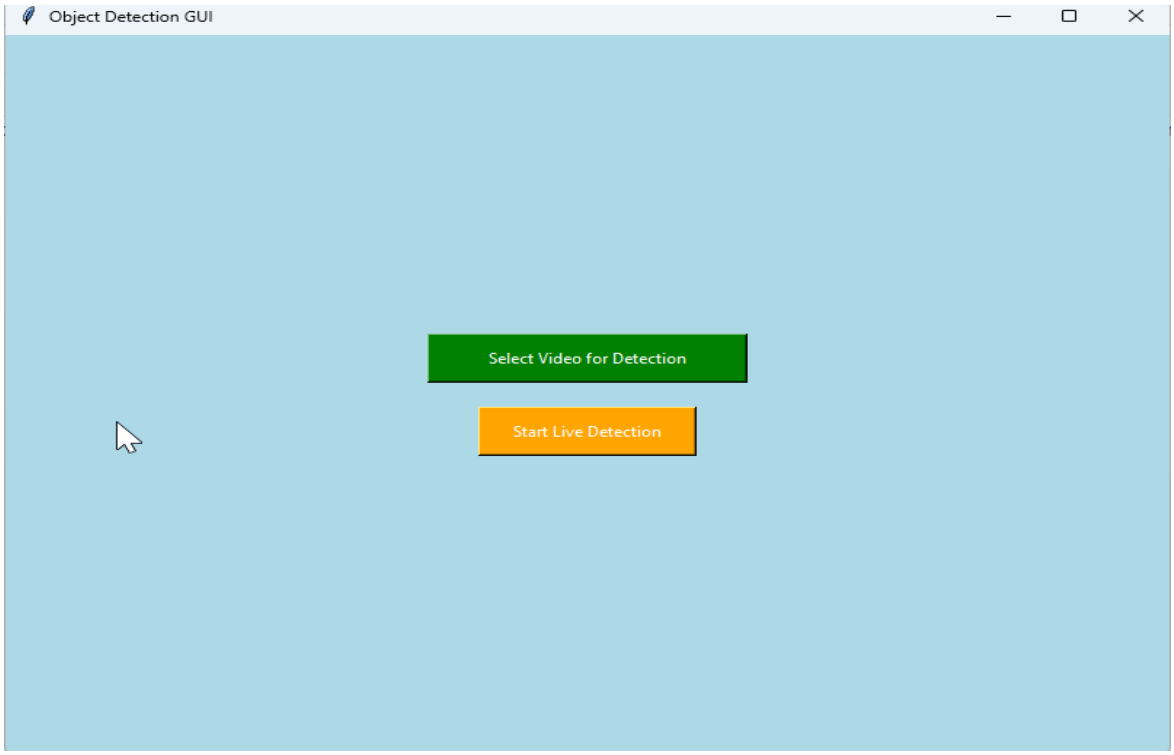
[8] Nath, Behzadan & Paal (2020) Nath ND, Behzadan AH, Paal SG. Deep learning for site safety: real-time detection of personal protective equipment. Automation in Construction. 2020;112:103085. doi: 10.1016/j.autcon.2020.103085.

[9] Zhang et al. (2021) Zhang C, Tian Z, Song J, Zheng Y, Xu B. Construction worker hardhat-wearing detection based on an improved BiFPN. 2020 25th international conference on pattern recognition (ICPR); Piscataway. 2021. pp. 8600–8607

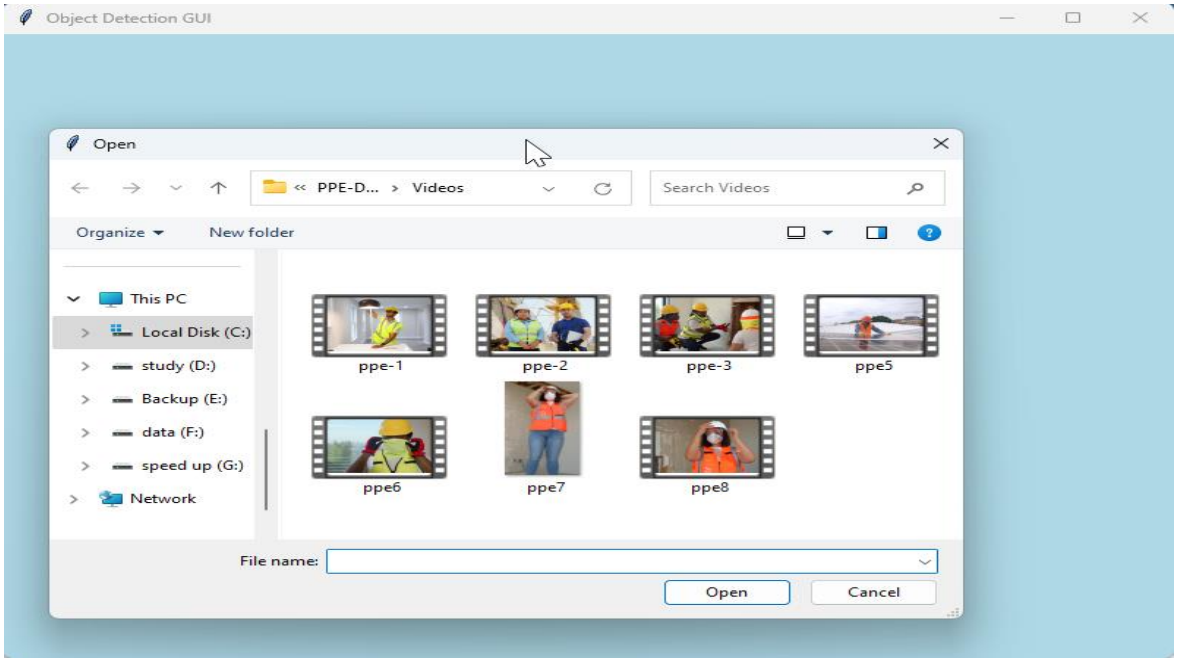
[10] Wang et al. (2021b) Wang Z, Wu Y, Yang L, Thirunavukarasu A, Evison C, Zhao Y. Fast personal protective equipment detection for real construction sites using deep learning approaches. Sensors. 2021b;21(10):3478. doi: 10.3390/s21103478.

Appendices

Appendix A: GUI – Interface

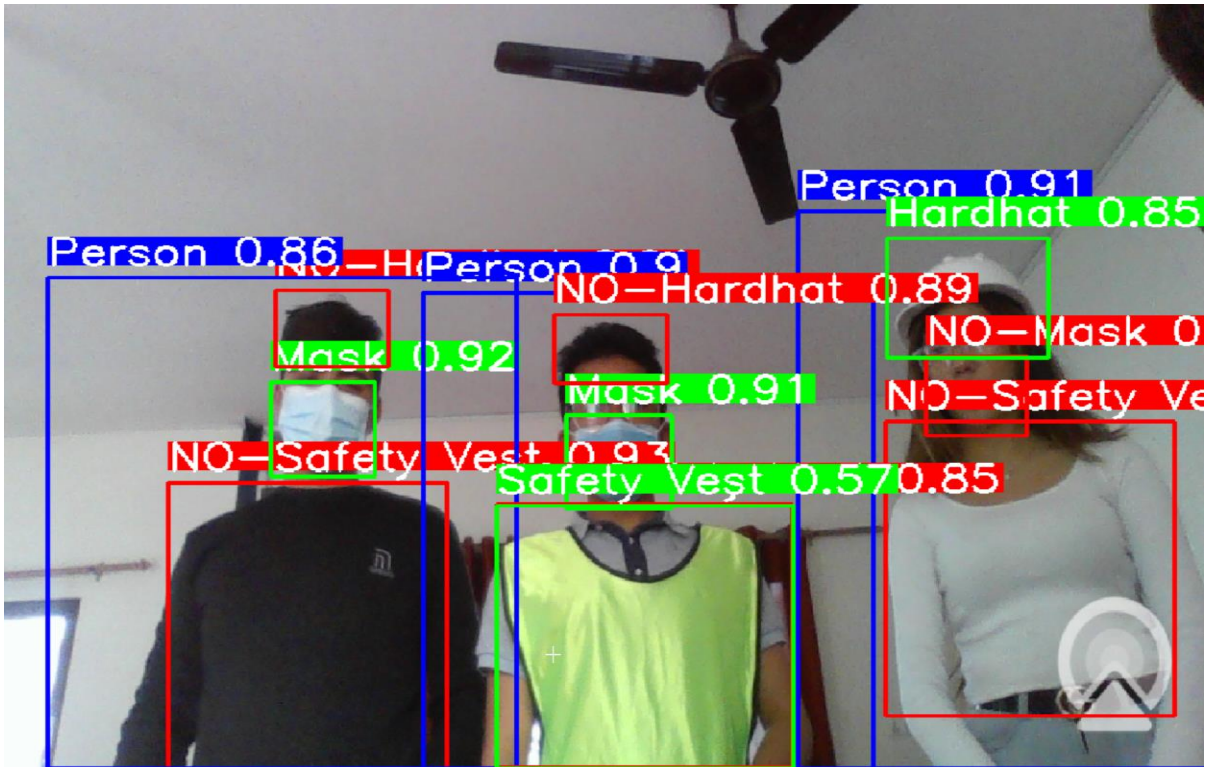


Appendix B: Selection of Video

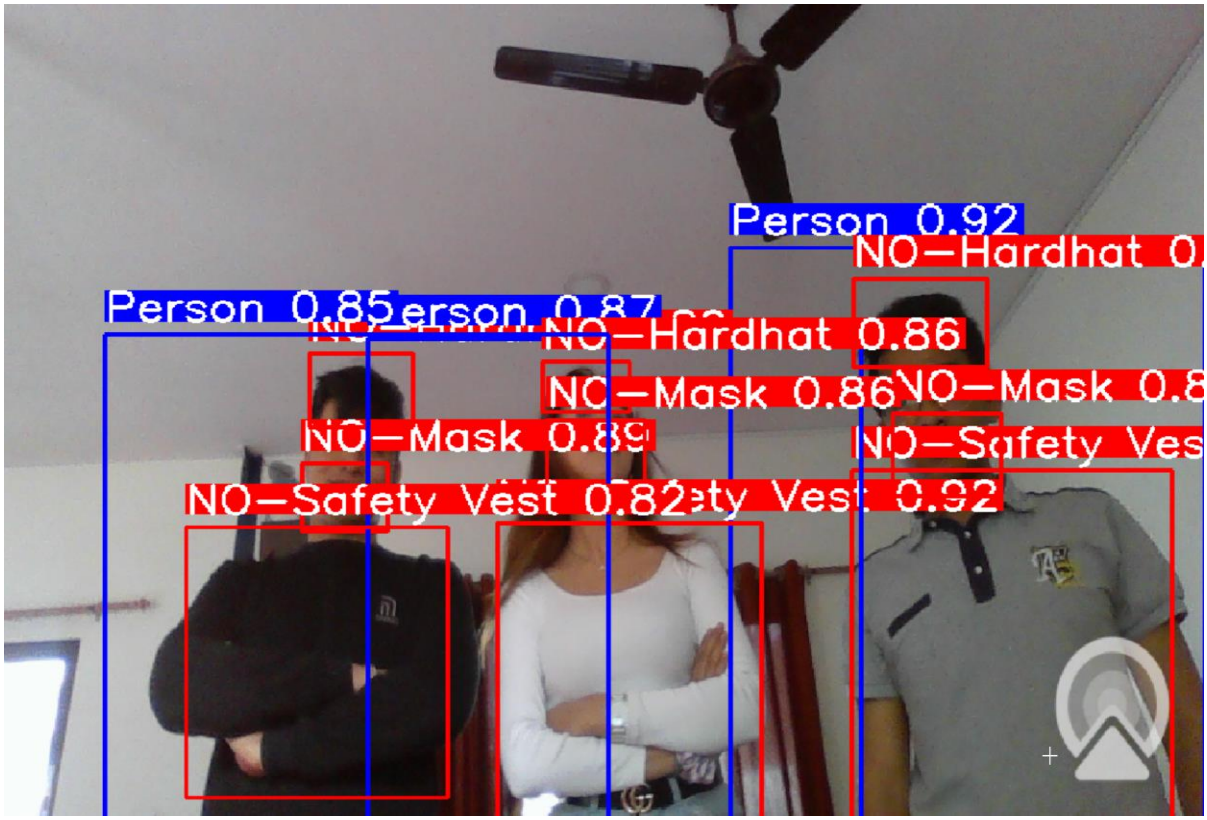


Appendix C: Detection being performed on video

Appendix D: Detection being performed real time




Hardhat 0.88
Mask 0.92
Safety Vest


Person 0.92
NO-Hardhat
Person 0.85
NO-Hardhat 0.86
NO-Mask 0.86
NO-Mask
NO-Mask
NO-Safety Ves
NO-Safety Vest 0.92