Workplace Safety Monitoring via Vision

*Github Repository: https://github.com/betulnesibe/workplace-safety-monitoring/*

Nesibe Betül Döner

Advisor: Prof. Dr. Oğuz Tosun

May 25, 2025

# Table of Contents

[1. Table of Contents 2](#_Toc199131666)

[2. Introduction 4](#_Toc199131667)

[3. Motivation 4](#_Toc199131668)

[4. Description of the System 5](#_Toc199131669)

[5. Requirements Specification 6](#_Toc199131670)

[5.1. Functional Requirements 6](#_Toc199131671)

[5.2. Non-Functional Requirements 6](#_Toc199131672)

[5.3. Dataset and Model Requirements 7](#_Toc199131673)

[6. Designed System 7](#_Toc199131674)

[6.1. System Architecture Overview 7](#_Toc199131675)

[6.2. Diagrams 8](#_Toc199131676)

[6.3. Design Details 10](#_Toc199131677)

[7. Implementation 11](#_Toc199131678)

[7.1. Model Selection and Dataset Preparation 11](#_Toc199131679)

[7.2. Base Training and Fine-Tuning 13](#_Toc199131680)

[7.3. Sample Inference and Visual Validation 14](#_Toc199131681)

[7.4. Hyperparameter Tuning and Fine-Tuning 14](#_Toc199131682)

[7.5. Test-Time Augmentation (TTA) 14](#_Toc199131683)

[7.6. K-Fold Cross-Validation 14](#_Toc199131684)

[7.7. Weighted Ensembling via Weighted Boxes Fusion (WBF) 15](#_Toc199131685)

[7.8. Knowledge Distillation (Student Model) 16](#_Toc199131686)

[7.9. Model/Method Selection 16](#_Toc199131687)

[7.10. Web Application Development 17](#_Toc199131688)

[8. Demonstration of functionality 18](#_Toc199131689)

[8.1. User Interface Overview 18](#_Toc199131690)

[8.2. Image Upload Mode 19](#_Toc199131691)

[8.3. Video Upload Mode 20](#_Toc199131692)

[8.4. Webcam Stream Mode 22](#_Toc199131693)

[8.5. Violation Reporting and Output 23](#_Toc199131694)

[9. Conclusion 23](#_Toc199131695)

[10. Future Work 24](#_Toc199131696)

[11. References 25](#_Toc199131697)

# Introduction

Ensuring compliance with occupational safety standards is a critical concern in industrial environments, where lapses can result in serious injuries, legal liabilities, or even fatalities. Traditionally, monitoring workplace safety, especially the use of Personal Protective Equipment (PPE)—has relied on manual inspection or periodic auditing. These approaches are inherently limited in scalability, prone to human error, and often fail to provide timely intervention.

With the increasing adoption of Industry 4.0 principles, software-based automation is being integrated into safety management systems. Among these, computer vision systems powered by deep learning have shown remarkable promise in object detection and activity recognition, making them well-suited for applications such as construction site PPE compliance monitoring. These systems can offer continuous, objective, and real-time assessments of workplace safety conditions, thereby augmenting traditional approaches. This project focuses on video-based PPE violation detection rather than real-time streaming due to computational constraints and the use case’s offline nature, making it suitable for post-incident analysis and periodic audits.

This project presents a modular, AI-based workplace safety monitoring application that focuses on PPE detection using image, video, and webcam input. While real-time streaming performance is limited due to computational constraints, the system is capable of handling live video with reduced speed. The proposed system utilizes state-of-the-art object detection models and a rule-based violation checker to detect non-compliance in uploaded or streamed content. The application prioritizes usability, performance, and future extensibility.

# Motivation

Ensuring worker safety is not only a legal requirement but also a moral and economic imperative. Workplace accidents can result in severe injuries, loss of productivity, and even fatalities. According to the International Labour Organization (ILO) (2015), over 2.3 million people globally die each year due to work-related accidents or diseases, and many of these incidents could be prevented through better compliance with safety measures.

Despite the availability of protective equipment, non-compliance remains a frequent issue. Workers may intentionally or unintentionally neglect wearing PPE, especially in fast-paced or unregulated environments such as construction sites, where risks are elevated. Traditional monitoring methods, such as manual inspection or supervisor oversight, are insufficient for continuous enforcement, particularly in large or dynamic workplaces. These limitations highlight the need for scalable, automated systems capable of monitoring safety compliance in real time or near real time.

This project is motivated by the potential of computer vision to bridge this gap. By automating PPE detection through image and video analysis, including live webcam feeds, organizations can significantly reduce their reliance on manual monitoring, improve compliance, and respond proactively to violations. Furthermore, by adopting a lightweight and open-source-friendly design, this system can serve as a foundational step toward broader digital transformation in safety management, particularly for small and medium-sized enterprises (SMEs) that lack resources for advanced automation.

# Description of the System

The selected system is an AI-driven workplace safety monitoring tool that specializes in detecting the presence or absence of Personal Protective Equipment (PPE) in workplace imagery. The system utilizes a fine-tuned object detection model (YOLOv8) to identify key PPE elements; helmets, masks, and safety vests on workers and other safety-critical items appearing in video frames or images, specifically within construction sites.

Unlike real-time surveillance systems that require complex infrastructure and high computational resources, this system mainly focuses on on-demand analysis. Users can upload a video, an image, or even use a webcam stream with limited capabilities captured in a workplace setting. The system analyses each frame or image to determine whether workers are in compliance with safety regulations. Any detected safety violation, such as a missing helmet or mask, is logged and visualized, providing an interpretable summary of compliance levels.

The prototype includes the following core components:

* Media Upload Interface: A user interface where users can select image, video files, or initiate a webcam stream.
* Frame Extraction & Preprocessing: Uploaded media or streamed frames are processed into frame units for analysis.
* PPE Detection Model: A pretrained YOLOv8 object detection model, fine-tuned on PPE datasets, processes each frame to identify safety gear.
* Violation Analysis Engine: A rule-based logic module evaluates model predictions and flags potential violations.
* Report Generation: A visual and textual report summarizing detected violations, including annotated frames and timestamped logs (for videos).

The solution is designed to be lightweight, platform-independent, and extendable, making it suitable for educational, experimental, and small-scale deployment scenarios.

# Requirements Specification

The system requirements are divided into three categories: functional requirements, non-functional requirements, and dataset/model requirements.

## Functional Requirements

* + 1. The system shall allow users to upload video and image files for analysis.
    2. The system shall allow users to perform real-time detection via webcam.
    3. The system shall extract frames from the uploaded video at configurable intervals.
    4. The system shall detect the presence of helmets, masks, and vests on individuals and other supported PPE classes within each frame.
    5. The system shall compare detected PPE objects with predefined compliance rules to identify safety violations.
    6. The system shall display a summary of violations with timestamps.
    7. The system shall present detection results through the user interface with visual feedback.

## Non-Functional Requirements

* + 1. The system shall operate as a local desktop or web application without requiring cloud deployment.
    2. The inference time for a 1-minute video (at 1 fps) shall not exceed 120 seconds on a mid-range machine.
    3. The system shall achieve at least 80% detection accuracy (mAP) for PPE compliance on test datasets.
    4. The user interface shall be minimalistic and require no technical expertise to operate.

## Dataset and Model Requirements

* + 1. The system shall use publicly available annotated datasets for training and evaluation.
    2. The object detection model shall be pretrained and fine-tuned for PPE classes.
    3. The model shall be evaluated using metrics such as precision, recall, mAP, and mAP50-95.
    4. The system shall include a validation mechanism to test performance against held-out video samples.

# Designed System

The designed system is a modular, component-based application that emphasizes extensibility, maintainability, and software quality attributes. It leverages a layered architecture, separating concerns into input handling, data transformation, inference, and reporting stages.

## System Architecture Overview

The system consists of the following core components:

* Frontend Upload Interface
  + Implemented using Streamlit for a lightweight and interactive web experience.
  + Provides three input modes: image upload, video upload, and real-time webcam stream.
  + Features configurable detection filters and confidence thresholds.
  + Displays real-time metrics and violation summaries.
* Media Preprocessing Module
  + Handles multiple input formats with standardised preprocessing.
  + Implements frame skipping (every 5th frame) for efficient video processing.
  + Maintains aspect ratio while resizing to 640x640 for model compatibility.
  + Includes video compression with configurable target size.
  + Manages colour space conversions (BGR to RGB)
* PPE Detection Model
  + Implements a fine-tuned YOLOv8n fold model selected from 5-fold cross-validation (specifically Fold 2).
  + Detects ten safety-related classes, including PPE compliance and violations.
  + Provides configurable confidence thresholds for detection filtering.
  + Supports real-time inference for webcam streams.
* Violation Analysis Engine
  + Implements rule-based violation detection with duplicate prevention.
  + Tracks violations with precise timestamps and visual evidence.
  + Provides both real-time and post-processing analysis.
  + Includes violation highlighting and visual indicators.
  + Generates comprehensive violation reports with timelines.
* Visualization Module
  + Implements colour-coded detection visualization with legend.
  + Provides real-time violation metrics and counters.
  + Includes timeline visualization for video analysis.
  + Features configurable class filtering.
  + Implements violation highlighting with semi-transparent overlays.
  + Supports export of processed media with annotations

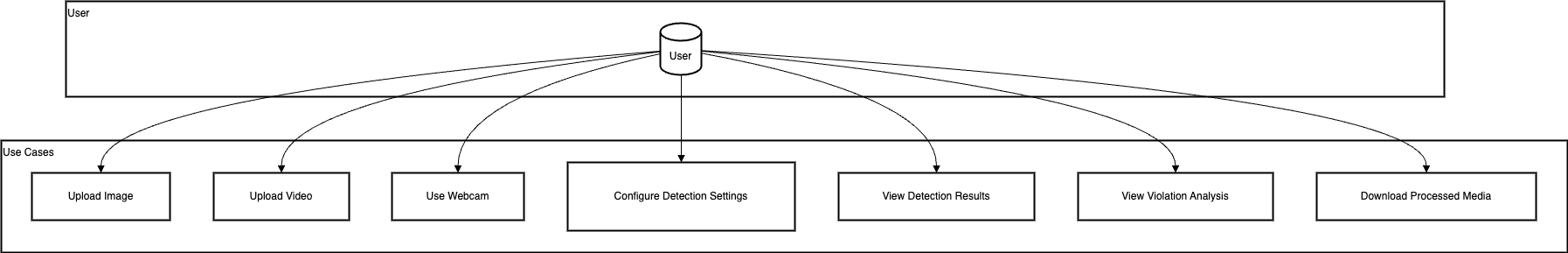
## Diagrams

* + 1. System Architecture Diagram

A diagram of a software process

AI-generated content may be incorrect.

* + 1. Use Case Diagram

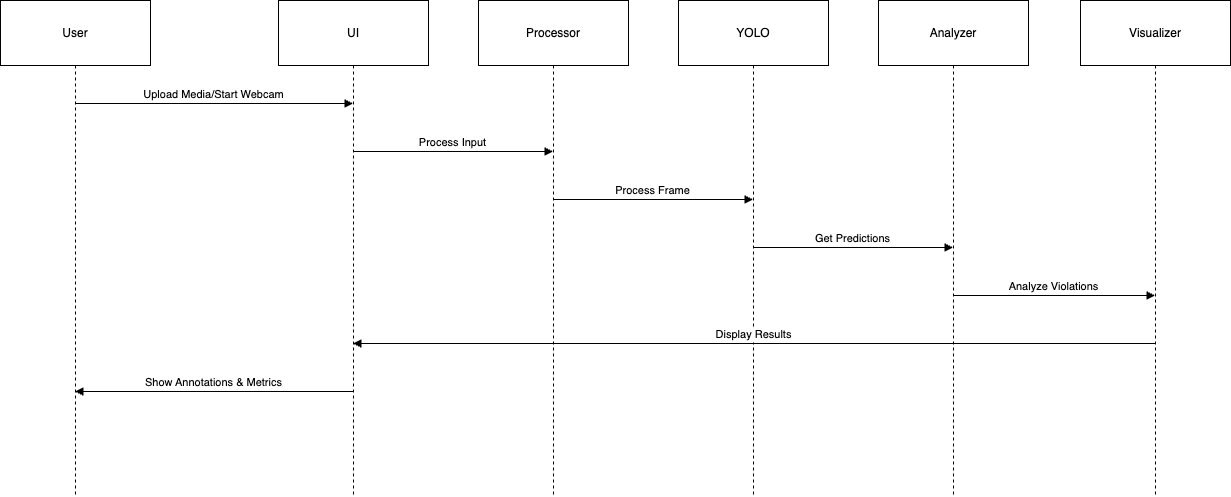


* + 1. Class Diagram

A diagram of a computer

AI-generated content may be incorrect.

* + 1. Sequence Diagram



* + 1. Activity Diagram

A diagram of process flow

AI-generated content may be incorrect.

## Design Details

* Modular Structure: Facilitates independent development, testing, and debugging of each component without affecting the rest of the system.
* Transfer Learning Approach: Reduces training time and hardware requirements while maintaining high detection accuracy by fine-tuning a pretrained YOLOv8 model.
* Flexible Input Modes: Supports not only on-demand video upload but also image upload and real-time webcam input, enabling broader usability in different environments.
* Scalability: The system architecture allows for future enhancements, such as full real-time stream handling, hazard recognition, or synthetic data generation.
* Usability: The user interface is designed with simplicity in mind, enabling non-technical users (such as site supervisors or safety officers) to use the tool without prior experience.

# Implementation

The implementation process followed a modular and systematic approach to developing a robust PPE detection system for construction site safety monitoring. It consisted of dataset preparation, model training, evaluation, hyperparameter tuning, test-time augmentation, cross-validation, ensembling, and knowledge distillation.

## Model Selection and Dataset Preparation

The system is based on the YOLOv8 object detection architecture, developed by Ultralytics (Ultralytics, 2023). YOLOv8 was selected due to its high accuracy, real-time inference capability, and lightweight structure compared to earlier versions such as YOLOv5. The initial model used was the YOLOv8n variant, which provides a good balance between inference speed and accuracy.

A publicly available PPE dataset, Construction Site Safety Image Dataset Roboflow (Sanyal, 2022) was used as the basis for training. This dataset includes annotated images of workers wearing or not wearing required safety equipment such as helmets, masks, and safety vests, as well as additional classes like person, vehicle, safety cone, and machinery. All annotations of the dataset were matching YOLO format and split into training, validation, and test sets.

A blue rectangular bar graph

AI-generated content may be incorrect.

*Data distribution of the dataset*

A graph showing the number of trains

AI-generated content may be incorrect.

*Class distribution of the training data*

A graph of a number of bars

AI-generated content may be incorrect.

*Class distribution of the validation data*

A graph of a bar graph

AI-generated content may be incorrect.

*Class distribution of the test data*

## Base Training and Fine-Tuning

Initial training was conducted using the pretrained YOLOv8n weights. The model was trained for 50 epochs with a 640×640 image size on the prepared dataset. This training phase served as a baseline and provided the foundation for subsequent fine-tuning and evaluation.

The trained model was evaluated on the test set, and results were assessed using standard object detection metrics: Precision, Recall, mAP@0.5, and mAP@0.5:0.95.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model/Method | Precision | **Recall** | **mAP50** | **mAP50-95** |
| Base | 0.803 | 0.668 | 0.720 | 0.431 |

A graph of a loss

AI-generated content may be incorrect.

*Training loss, precision, recall, and mAP metrics across 50 epochs during base model training*

A graph of safety vests

AI-generated content may be incorrect.

*Confusion matrix*

## Sample Inference and Visual Validation

To qualitatively verify detection outputs, sample video frames were processed using the trained model. Annotated images were generated to visualize bounding boxes and predicted classes. These visualizations helped identify strengths and weaknesses of the model, especially in challenging scenarios such as partial occlusion or cluttered scenes.

## Hyperparameter Tuning and Fine-Tuning

The base model was further refined through hyperparameter tuning. A custom training configuration was prepared, adjusting key parameters such as the learning rate, momentum, augmentation strategies, and warm-up behaviour. The model was fine-tuned for an extended number of epochs, using the previously trained weights as a starting point.

This fine-tuned model, referred to as yolov8n-tuned2, demonstrated improved accuracy and generalization compared to the initial training.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model/Method | Precision | **Recall** | **mAP50** | **mAP50-95** |
| yolov8n-tuned2 | 0.857 | 0.710 | 0.759 | 0.467 |

## Test-Time Augmentation (TTA)

To further enhance robustness, Test-Time Augmentation (TTA) was applied during evaluation. This involved running multiple augmented variants of each test image through the model and aggregating the predictions. TTA slightly improved performance by reducing false negatives and increasing detection stability, especially for small or partially visible PPE items.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model/Method | Precision | **Recall** | **mAP50** | **mAP50-95** |
| TTA | 0.842 | 0.698 | 0.759 | 0.475 |

## K-Fold Cross-Validation

To validate the model’s generalization, 5-fold cross-validation was performed on the fine-tuned model. The training set was split into five subsets, and five separate models were trained and evaluated using rotating validation sets with each fold used once as the validation set while the remaining four served as the training data.

Performance metrics for each fold were recorded and compared. This approach helped ensure that the model’s performance was not overly dependent on a specific data subset. While all fold models achieved satisfactory performance, results showed slight variations in class-wise performance due to dataset differences across folds.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model/Method | Precision | **Recall** | **mAP50** | **mAP50-95** |
| Fold0 | 0.863 | 0.698 | 0.759 | 0.475 |
| Fold1 | 0.857 | 0.711 | 0.759 | 0.459 |
| Fold2 | 0.887 | 0.702 | 0.766 | 0.472 |
| Fold3 | 0.830 | 0.705 | 0.756 | 0.463 |
| Fold4 | 0.858 | 0.700 | 0.755 | 0.459 |

## Weighted Ensembling via Weighted Boxes Fusion (WBF)

To combine the strengths of each fold model, Weighted Boxes Fusion (WBF) was used to create an ensemble predictor. WBF works by merging overlapping predictions from multiple models into a single output, taking confidence scores into account. This method was preferred over Non-Maximum Suppression (NMS) for its ability to preserve high-confidence predictions from multiple sources.

However, although preliminary testing suggested strong performance, further detailed evaluation showed that the ensemble model had lower-than-expected metrics (precision: 0.76, recall: 0.77) compared to individual folds. This prompted a reevaluation of the final model selection.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model/Method | Precision | **Recall** | **mAP50** | **mAP50-95** |
| Ensemble (WBF) | 0.761 | 0.774 | NA | NA |

## Knowledge Distillation (Student Model)

To reduce inference cost while maintaining ensemble-level accuracy, a student model was trained using knowledge distillation. The ensemble model acted as a teacher, and its predictions were used to guide the student model during training by changing labels of the dataset. The resulting model is found to be not better than the previous models//methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model/Method | Precision | **Recall** | **mAP50** | **mAP50-95** |
| Knowledge Distillation | 0.845 | 0.670 | 0.730 | 0.424 |

## Model/Method Selection

To determine the most suitable model for deployment, several versions were trained, validated, and evaluated on the test set using consistent metrics: Precision, Recall, mAP@0.5, and mAP@0.5:0.95. These models included the base YOLOv8n model, a fine-tuned version, five fold-specific models from k-fold cross-validation, a weighted ensemble model, and a distilled (student) model trained using ensemble predictions.

While the ensemble initially appeared to perform best, a later detailed evaluation revealed inconsistencies in its results. Upon recalculating, the ensemble model achieved 0.76 precision and 0.77 recall, which were lower than expected. In contrast, Fold 2 from cross-validation achieved consistently high accuracy, recall, and mAP scores across the full test set.

Thus, Fold 2 was selected as the final deployed model based on its strong overall performance, stable class-wise results, and suitability for deployment.

A table with numbers and a number

AI-generated content may be incorrect.

*Precision, Recall, and mAP scores of all evaluated model variants on the test set.*

A table with numbers and a number of numbers

AI-generated content may be incorrect.

*Overall and class mAP@0.5:0.95 scores of all evaluated model variants on the test set.*

## Web Application Development

The PPE detection system was implemented as an interactive Streamlit web application with three main input modes: image upload, video processing, and real-time webcam monitoring.

### Core Components

* User Interface: Tabbed interface with configurable detection settings and real-time visualization.
* Final Model: YOLOv8n model trained on Fold 2 of cross-validation.
* Processing Pipeline: Optimized frame processing with automatic resizing (640x640) and frame skipping (every 5th frame).
* Visualization Engine: Color-coded annotations, violation highlighting, and real-time metrics.

### Key Features

* Detection Configuration:
  + Adjustable confidence threshold (0.0 to 1.0).
  + Class filtering for PPE compliance and safety elements.
  + Violation-only mode for focused monitoring.
* Performance Optimizations:
  + FFmpeg-based video compression with H.264 codec.
  + Efficient memory management and resource cleanup.
  + Automatic display size adjustment.
* Error Handling:
  + Input validation for supported formats.
  + Graceful degradation for playback issues.
  + Comprehensive user feedback.

The implementation prioritizes performance and user experience while maintaining detection accuracy and providing comprehensive safety violation monitoring capabilities.

### Output and Reporting

* Real-time Visualization:
  + Color-coded bounding boxes (green for compliance, red for violations).
  + Confidence scores and class-specific annotations.
  + Semi-transparent overlays for violation frames.
* Violation Analysis:
  + Frame-wise violation counting with duplicate filtering.
  + Timestamp-based violation logging (MM:SS format).
  + Violation timeline charts and real-time counters.
* Export Capabilities:
  + Downloadable annotated videos with H.264 compression.
  + Exportable violation timelines and metrics.
  + Frame-wise detection results with timestamps.

# Demonstration of functionality

## User Interface Overview

The system is implemented as a lightweight web application using Streamlit, offering an interactive, browser-based interface. Designed for local execution within a standard Python environment, it allows users to:

* Select an input mode: image upload, video upload, or webcam stream.
* Adjust detection settings such as confidence threshold and class filters (including a violation-only view).
* View annotated results with bounding boxes and visual violation highlights.
* Export processed media (e.g., annotated video) for further documentation.

## Image Upload Mode

In this mode, users can upload a single image file (JPG, JPEG, PNG). Once uploaded, the system automatically performs inference using the YOLOv8n Fold 2 model. Detected objects, including both compliant and non-compliant PPE and safety elements, are displayed with color-coded bounding boxes and class labels. Frames containing safety violations (NO-Hardhat, NO-Mask, NO-Safety Vest) are visually highlighted with a semi-transparent overlay.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Video Upload Mode

For video files (MP4, AVI), the system processes selected frames (currently every 5th frame for efficiency). Frame-by-frame detection is performed by the Fold 2 model. Users can view:

* Annotated video playback within the interface.
* A violation timeline chart showing the distribution of violations across processed frames.
* Timestamped violation logs (MM:SS format) detailing when specific safety infractions occurred.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Webcam Stream Mode

The system supports basic real-time video input from the user’s default webcam. Upon activating this mode, frames are captured live and passed through the Fold 2 model with minimal preprocessing. Detected objects and violations are visualized immediately on the screen. Real-time counters display the cumulative number of detected hardhat, mask, and safety vest violations during the stream. Performance may vary depending on the user’s hardware capabilities.

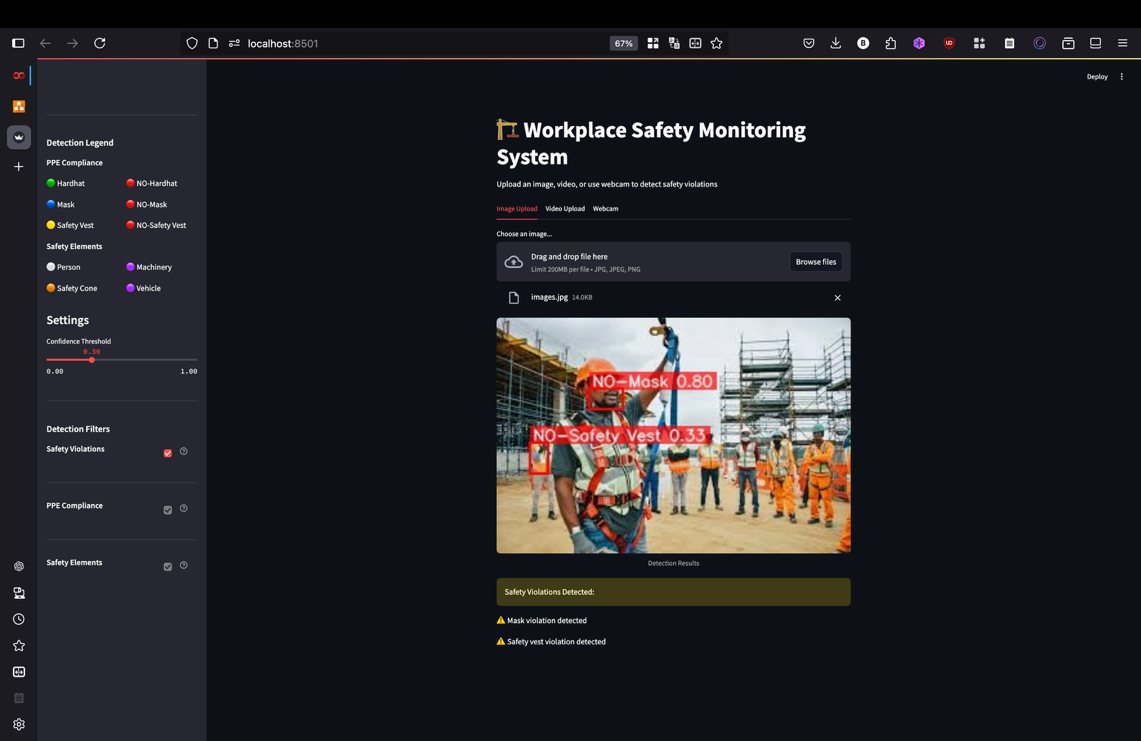
A screenshot of a computer

AI-generated content may be incorrect.

## Violation Reporting and Output

Each detection mode generates visual and textual outputs summarizing the analysis, with a strong focus on safety violations. These include:

* Annotated images and videos with color-coded bounding boxes and labels.
* Visual highlighting of frames containing violations.
* For video, a violation timeline chart and timestamped logs for NO-Hardhat, NO-Mask, and NO-Safety Vest detections.
* Real-time violation counters for webcam analysis.
* Export functionality to download the processed annotated video.



# Conclusion

This project demonstrated the development of a computer vision-based workplace safety monitoring system that detects PPE compliance using images, video recordings, and webcam streams. The final application targets construction environments, where non-compliance can lead to severe consequences, and focuses specifically on detecting hardhats, masks, safety vests, and related safety elements.

Throughout the implementation, a fine-tuned YOLOv8n model trained via k-fold cross-validation (specifically Fold 2) was selected as the final deployed model due to its balanced and robust performance. Various enhancements were integrated during the model development pipeline, including hyperparameter tuning, test-time augmentation (TTA), and knowledge distillation. While ensembling via Weighted Boxes Fusion (WBF) was explored, further analysis showed that Fold 2 yielded better and more consistent results for deployment.

The web-based interface developed using Streamlit allows for user-friendly interaction, even for non-technical users, with support for multiple input modes and real-time visualization. Frame-wise violation reporting, timestamped summaries, and exportable outputs make this system useful not only for immediate detection but also for post-incident review and training purposes.

# Future Work

Several areas have been identified for improvement and future expansion:

* Real-Time Performance Optimization: While the system supports webcam input, real-time performance could be significantly improved by model quantization or conversion to TensorRT or ONNX for GPU acceleration.
* Hazard Detection: In addition to PPE, future versions could include fall detection, unsafe zone monitoring, or machinery proximity alerts.
* Deployment Scaling: With additional optimization, this tool can be integrated into edge devices (e.g., smart cameras or Raspberry Pi) for real-time, on-site use.
* Synthetic Data Augmentation: To improve class balance and generalization, future experiments may incorporate synthetic data generated via tools like Unity or Blender.
* Multi-Camera Support: Expanding the system to simultaneously monitor multiple cameras can enable broader safety coverage across construction sites.

This project lays a practical and technical foundation for AI-assisted safety monitoring and demonstrates how accessible tools and open-source models can be leveraged for social benefit, especially for under-resourced environments like SMEs in construction.

# References

Anthropic. (2025). Claude AI [Constitutional AI assistant]. Available at: https://claude.ai

International Labour Organization. (2019). Safety and health at the heart of the future of work: Building on 100 years of experience (p. 6). <https://webapps.ilo.org/static/english/osh/en/story_content/external_files/fs_st_1-ILO_5_en.pdf>

OpenAI. (2025). ChatGPT (GPT-4o) [Large language model]. Available at: <https://chat.openai.com>

Sanyal, S. (2022). Construction Site Safety Image Dataset. Retrieved from <https://www.kaggle.com/datasets/snehilsanyal/construction-site-safety-image-dataset-roboflow>

Ultralytics. (2023). YOLOv8: <https://docs.ultralytics.com>