EDISS Winter School 2025 - Hackathon Report

Real-Time PPE Detection for Workplace Safety

Mete Harun Akcay

March 17, 2025

1 Introduction

Ensuring the proper use of Personal Protective Equipment (PPE) is crucial in workplaces such as construction sites, warehouses, and manufacturing facilities. Non-compliance with PPE regulations can result in serious consequences, including accidents, financial losses, and legal penalties [1]. Approximately 60% of workers consistently wear PPE while on duty [1]. For the remaining workforce, it is crucial to conduct regular PPE checks to ensure compliance and maintain workplace safety. However, many industries still rely on manual PPE checks, which are often slow, error-prone, and difficult to scale, especially in large industrial environments [2].

To address these challenges, AI-driven PPE detection systems have emerged as a promising solution. However, existing AI solutions frequently struggle with scalability, efficiency, and the ability to deliver real-time alerts, limiting their practical application in dynamic industrial settings [2].

In response to these limitations, we developed a computer vision-based PPE detection system designed to provide real-time, scalable, and efficient monitoring. Our solution leverages a robust architecture that integrates advanced technologies to ensure safety compliance with instant alerts and low-latency processing, making it suitable for large-scale industrial environments. This system is designed to not only improve detection accuracy but also enhance operational efficiency and workplace safety. To provide a comprehensive overview of our solution, the remainder of this report is structured as follows:

- In Section 2, we describe our approach, including the detailed system architecture, the chosen technology stack, and the data collection & modeling process.
- Section 3 outlines the implementation of the system, covering the front-end design, the back-end and microservice architecture, and our storage strategy.
- In Section 4, we present the system's performance results along with a discussion of key insights.
- Finally, Section 5 reflects on the hackathon experience, highlighting key challenges, feedback to supervisors, and recommendations for future winter schools.

2 Our Approach

2.1 System Architecture

Our PPE detection system follows a modular architecture designed to support real-time performance, scalability, and efficient monitoring. The system integrates multiple components that work together to deliver accurate PPE detection and timely notifications.

The system's data flow begins with video streams collected from cameras, which are processed through a **Kafka cluster**. Kafka handles real-time data streaming and distributes the video feeds to various services. A **Gateway Coordinator** is responsible for managing core functionalities, including data processing, annotated video streaming, notification delivery, and PPE detection using YOLOv8.

The backend is built using **Flask** and follows a microservice architecture, where each service is independently managed but interconnected through the Kafka cluster. The frontend, developed with **ReactJS** and **Material UI**, provides users with a clear interface to monitor live video feeds, view detection results, and receive notifications.

For data storage, the system combines **Redis Cache** for managing active user sessions, **MongoDB** for structured data storage, and **Elasticsearch** integrated with **Kibana** for logging and visualizing system activity.

Figure 1 provides an overview of these components and their interactions.

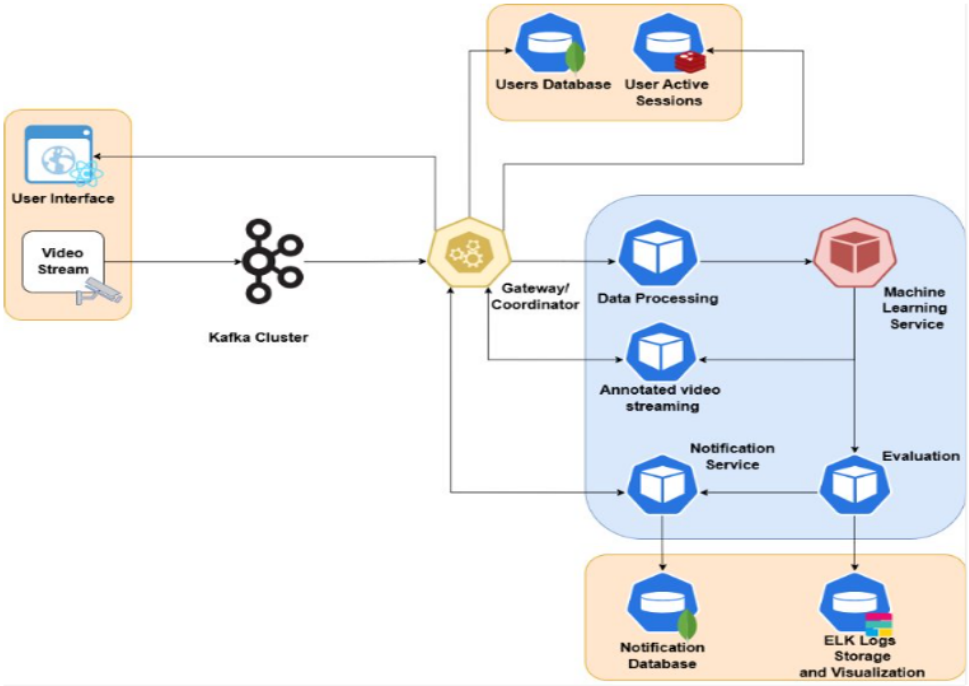

Figure 1: Overview of the system architecture.

2.2 Technology Stack

Our system leverages a combination of tools and frameworks carefully selected to ensure efficient development, scalability, and performance. Each component plays a distinct role in supporting the system's architecture.

Kafka was chosen as the core message broker to manage real-time video stream distribution. Its ability to handle high-throughput data with low latency makes it ideal for efficiently transmitting camera feeds to various services. Kafka's scalability also ensures the system can accommodate multiple cameras without performance degradation.

YOLOv8 was selected as the object detection model due to its strong balance between speed and accuracy. Its lightweight architecture makes it particularly suitable for real-time inference, a crucial requirement for PPE detection. If we had more time, other models such as SSD could have been tested as well.

For the backend, we employed **Flask** to develop a microservice-based architecture. Flask's flexibility allowed us to create modular services that could be easily integrated with Kafka and other system components.

On the frontend, we used **ReactJS** alongside **Material UI** to build an intuitive and user-friendly interface. This combination allowed us to create a dashboard where users can monitor PPE detection results in real-time, view compliance statistics, and receive alerts.

To manage data efficiently, we integrated multiple storage solutions. **Redis** was decided to be used to manage active user sessions and enable fast socket-based notifications. **MongoDB** was employed for storing structured data such as user profiles, camera information, and notification records, providing flexibility

in handling dynamic data structures. Additionally, we decided to incorporate **Elasticsearch** along with **Kibana** to store system logs and visualize insights about detections, compliance rates, and alert trends.

Finally, for the annotation process, we used **LabelImg**, a versatile tool that allowed us to manually label PPE items in the provided dataset. This ensured the training data was accurately prepared for YOLOv8.

3 Implementation

3.1 User Interface

Our system's user interface was designed to provide users with a clear and intuitive way to monitor PPE detection results, track compliance rates, and respond to alerts efficiently. The interface is divided into three main sections: the dashboard, the camera view, and the statistics page.

The **Dashboard** offers an overview of key metrics, including the total number of detected PPE usages, the overall compliance rate, and detailed PPE alerts. Users can view individual alerts to inspect specific compliance issues.

The **Camera View** enables users to monitor live video streams from selected cameras. This section also provides real-time notification alerts that highlight detected PPE violations. Key metrics such as total detections, alerts, and compliance rates are displayed alongside the video feed to give users immediate insights.

The **Statistics Page** visualizes PPE data trends over various timeframes, including weekly, monthly, and yearly statistics. Users can explore breakdowns of PPE usage by type to identify patterns in compliance behavior. Additionally, this section allows users to download the data as a CSV file for further analysis.

Sample pages from the user interface are shown in Figure 2.

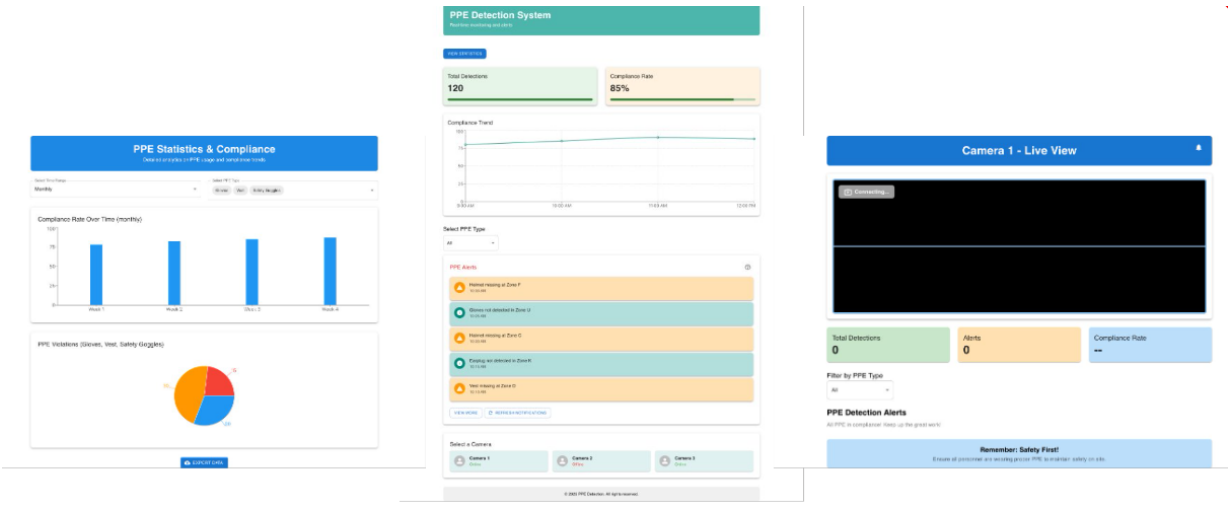

Figure 2: Sample pages from the user interface.

3.2 Backend and Storage

The backend of our system is built using **Flask** and follows a microservice architecture to ensure scalability and facilitate easier improvements in system performance. This design enables independent management of core services while supporting load balancing. The backend is connected to a **Kafka cluster**, which serves as the primary data stream manager. The app's entry point subscribes to Kafka topics, ensuring efficient data flow across services.

For data storage, we employed a multi-layered approach to handle various system needs. **Redis** is used as a fast, in-memory cache to manage active user sessions and support real-time socket-based notifications. For structured data storage, we utilized **MongoDB**, which stores user information, camera details, and notification records. Additionally, we integrated **Elasticsearch** to manage system logs and provide comprehensive visual insights through **Kibana**.

3.3 Data Collection & Modeling

We were provided with a dataset consisting of 536 images containing PPEs or workers with or without PPEs by the hackathon organizer. We removed the duplicate images, resulting in a final set of 445 unique images.

To prepare the dataset for training, we resized all images to 640x640, a suitable input size for YOLOv8 training. Additionally, since the images were unlabeled, we manually annotated them using the `LabelImg` tool. The annotation process included labeling the following nine classes: Helmet, Vest, Safety_Goggles, Hearing_Protection, Hairnet, Gloves, Coat, Overshoes, Worker. Figure 3 shows an example of a worker and the corresponding annotation.

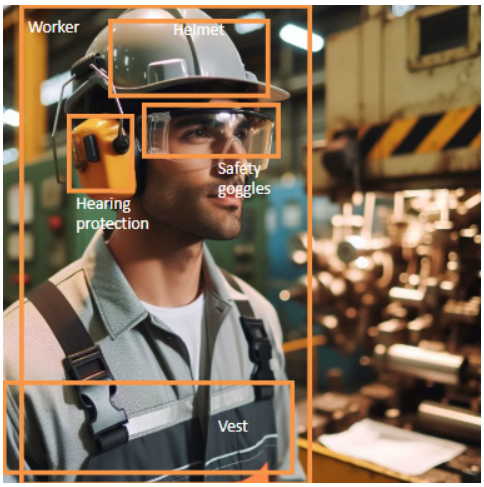

Figure 3: Example of a worker image with annotated PPE labels.

The annotated dataset was then split into three subsets with an 80/10/10 ratio for training, validation, and testing, respectively. For model training, YOLOv8 was employed with 100 epochs on a Tesla T4 GPU.

4 Results & Discussion

Table 1 presents the validation performance summary of our YOLOv8 model across various PPE classes. The model achieved promising results in detecting certain classes but also faced challenges due to data limitations.

Table 1: Validation Performance Summary

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	42	99	0.654	0.522	0.599	0.228
Helmet	11	12	0.622	0.687	0.766	0.282
Vest	3	4	0.378	0.250	0.285	0.114
Safety_Goggles	13	14	0.899	0.635	0.648	0.317
Hearing_Protection	18	26	0.567	0.654	0.658	0.176
Hairnet	19	28	0.610	0.536	0.577	0.209
Coat	8	15	0.847	0.371	0.660	0.270

As shown in the table, the model performed particularly well in detecting **Helmet**, **Safety Goggles**, and **Hearing Protection**, achieving relatively high precision and recall scores. The detection of **Hairnet** showed moderate performance, while the system struggled to identify **Vest** and **Coat**, often misclassifying them as background.

The confusion matrix (Figure 4) reveals that the model had difficulty distinguishing between vests and background objects, which may be attributed to the varying colors of vests present in the dataset. Additionally, the confusion matrix showed no instances of **Overshoes** or **Gloves**, since there were no corresponding examples in the validation dataset.

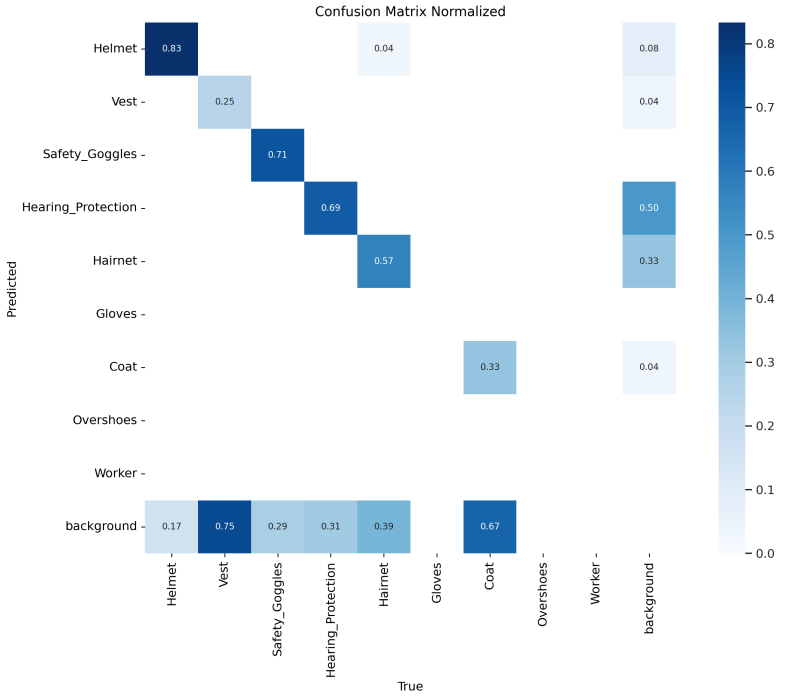

Figure 4: Confusion matrix with normalized values.

To further evaluate the model's behavior, Figure 5 presents the training and validation curves. The graph displays metrics such as **box loss**, **classification loss**, and mAP scores across epochs.

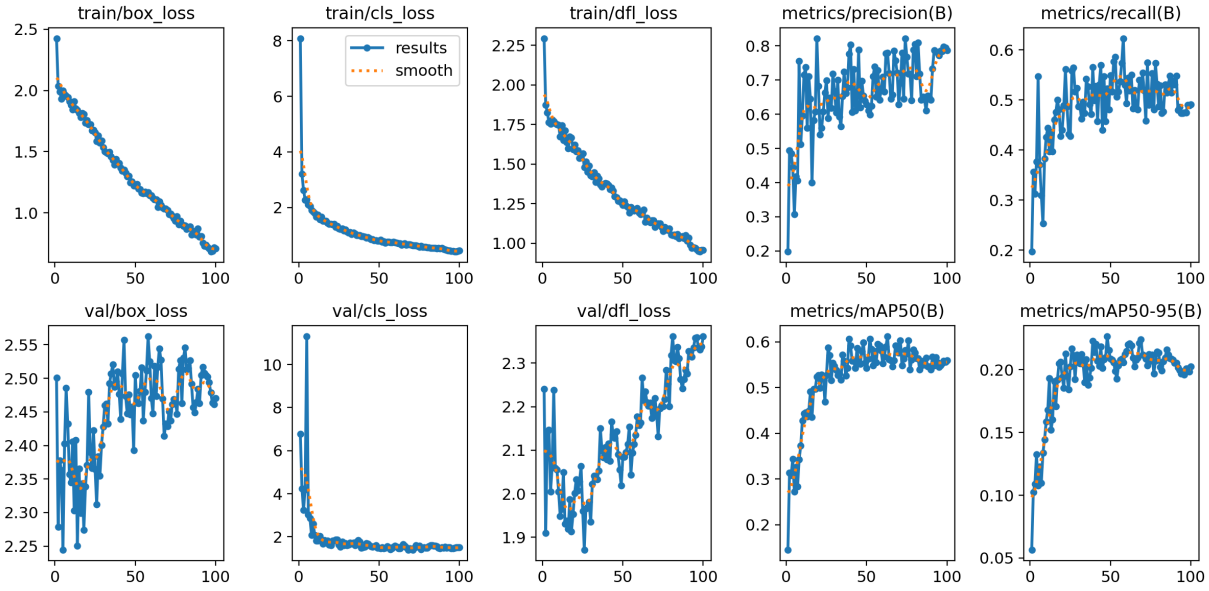

Figure 5: Training and validation performance curves.

The results indicate that the model gradually improved throughout training. The mAP@50 score reached approximately 0.599, while the mAP@50-95 score remained lower at 0.228, reflecting the challenge of achieving high performance across all IoU thresholds. For further insights into the training process and individual metric trends, readers may refer to Figure 5.

Overall, while the model showed promising performance in detecting certain PPE classes, its limitations are closely tied to data availability and quality. Expanding the dataset, particularly with more diverse vest samples and additional examples of underrepresented classes such as **Overshoes** and **Gloves**, would likely improve performance.

Figure 6 shows 16 sample predictions from the validation set. These examples further illustrate the

model's strengths and weaknesses. The model demonstrated strong performance in distinguishing between **hairnets** and **helmets**, as well as accurately detecting **hearing protection** and **safety goggles**. However, consistent with the quantitative results, the model occasionally struggled to identify **vests**, reinforcing the earlier observation that vest detection was a notable challenge.

Figure 6: Sample predictions from the validation set.

To sum up, in this project, we successfully implemented a functional user interface that allows users to monitor PPE detection results, track compliance rates, and receive notifications. We also trained a YOLOv8 model for PPE detection, which performed well in identifying several PPE items despite challenges with certain classes. While the model and UI were functional, other planned features were not fully implemented due to time constraints. These included a comprehensive alerting system, improved scalability mechanisms such as load balancing, and enhanced data storage strategies for managing large-scale deployments. Nevertheless, the system's architecture was designed with these elements in mind, ensuring a solid foundation for future development to create a more robust and effective PPE detection solution.

The source code for this project is available on GitHub.

5 Reflection

Overall, I must say that I didn't enjoy this hackathon experience. The nature of the project itself felt more suited for a timeline of several weeks or even months, not just two half-days. Given the complexity of the task, the limited time we had was far from sufficient to deliver a well-rounded solution.

One major frustration was the provided dataset — it was unlabeled, contained numerous duplicates, and included poor-quality data. As a result, we spent almost the entire first day just doing manual labeling, which significantly limited the time we had for model development and system integration.

Another issue I noticed was the unbalanced group divisions. From my observations, most teams heavily relied on Intake3 students, who generally had more experience and seemed to lead their teams. Unfortunately, my team didn't have any students from Intake 3, which made things even more challenging. On top of that, some teams had members who were already doing internships at the company that organized the hackathon, and coincidentally (!), their internship project aligned with the hackathon task. Naturally, they had a huge advantage, and it was no surprise that their project outperformed the rest.

What felt particularly unfair was seeing some Intake4 students from these well-supported teams, spending their last two days enjoying Italy while still walking away with rewards. Meanwhile, my team of just four Intake 4 students worked hard throughout the event, yet faced an uphill battle from start to finish.

For future winter schools, I would strongly recommend either skipping the hackathon altogether or, if a hackathon is included, making it a purely theoretical, idea-based challenge or a coding-focused task that runs across the entire week. This would allow participants to engage more meaningfully with the project and contribute in a fairer environment.

References

[1] R. Sehsah, A. El-Gilany, and A. Ibrahim, "Personal protective equipment (ppe) use and its relation to accidents among construction workers," *Med Lav*, vol. 111, no. 4, pp. 285–295, 2020.

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