Article

PPE Recognition System Based on Improved YOLOv11-Safety Algorithm and Frontend-Backend Architecture Integration

Lu Bei [1],* and Joan Lazaro [1]

[1] University of the East, Manila, Philippines
* Correspondence: Lu Bei, University of the East, Manila, Philippines

Abstract: With the expansion of global urbanization, the construction industry, a pillar of the national economy, remains a high-risk sector. Although the number of work safety accidents and fatalities in China decreased year-on-year in 2024, housing construction projects still have high accident and fatality rates. Personal Protective Equipment (PPE) is the last effective defense for workers, but PPE compliance management faces challenges such as limited manual supervision and complex working environments. Leveraging advancements in AI and computer vision, this research designs an AI-driven deep learning system centered on the YOLOv11-Safety algorithm to realize real-time and accurate identification of 9 categories of PPE-related targets on construction sites. The system integrates Spring Boot (backend), Vue.js (frontend), and Flask (AI middleware) to build a practical intelligent safety supervision solution. Through improvements in feature extraction, feature fusion, loss function, and inference optimization, the YOLOv11-Safety algorithm achieves higher detection accuracy, faster inference speed on edge devices, and stronger robustness. The system promotes the digital transformation of construction site safety management from passive response to active prevention, providing an effective intelligent solution for workplace safety supervision.

Keywords: Improved Algorithm; YOLOv11-Safety; PPE Recognition System; Frontend-Backend Architecture

Received: 12 December 2025
Revised: 20 January 2026
Accepted: 04 February 2026
Published: 11 February 2026

1. Introduction

With the expansion of global urbanization, the construction industry, a national economic pillar, remains a high-risk sector. The ILO reports it employs 7% of the global workforce but accounts for 30% of major fatal accidents. In China, 2024 work safety accidents and fatalities dropped year-on-year, with construction industry incidents continuing to decline, though housing construction projects still have high accident and fatality rates among related sectors. Personal Protective Equipment (PPE) is the last effective defense for construction workers, as correct use can significantly reduce injury risks, yet PPE compliance management faces challenges including manual supervision limitations, complex dynamic work environments, and frequent concealed violations [1].

Recent advancements in AI and Computer Vision, especially deep learning-based object detection represented by the YOLO series, provide solutions to these challenges [2]. This research aims to design an AI-driven deep learning system, using YOLOv11-Safety as the core to realize real-time and accurate identification of 9 classes of PPE-related targets on construction sites. The system integrates Spring Boot (backend), Vue.js (frontend), and Flask (AI middleware) to build a practical intelligent safety supervision solution, promoting the digital transformation of construction site safety management from passive response to active prevention [3].

2. Literature Review on Domestic and International Research

Deep learning has significantly advanced personal protective equipment (PPE) detection, with convolutional neural network (CNN)-based algorithms, particularly the YOLO series, playing a central role. Despite these advancements, current research still faces several challenges, including low detection accuracy in complex environments, poor performance for small or occluded targets, difficulties balancing real-time processing with computational efficiency, and the absence of universal models capable of adapting to diverse scenarios [4]. To address these issues, the proposed system adopts YOLOv11, integrates Spring Boot and Vue to implement a front-end-back-end separated web system, and optimizes user interaction, enhancing both performance and usability.

Key deep learning studies on PPE and helmet detection demonstrate notable progress. An Automatic Helmet Violator Detection System combining YOLO and Mask R-CNN achieved helmet detection accuracy of 98.2% and number plate segmentation accuracy of 96.4%. An industrial PPE compliance framework based on YOLOv7 outperformed existing state-of-the-art methods, while a web-based YOLOv5 system for occupational safety reporting achieved a mean average precision (mAP) of 91.5% and overall accuracy of 98.3%, though its performance under low-light conditions remained limited [5].

Optimized YOLO-based algorithms tailored to specific scenarios further enhance detection capabilities. A lightweight DWFL-YOLO model (YOLOv8-based) demonstrated higher mAP with fewer parameters, while the MH-YOLO model (YOLOv8-based) applied to coal mine helmet detection achieved high mAP50 with a 10 ms detection time. A CPM-YOLO model (YOLOv11-based) designed for road helmet detection improved mAP@0.5 by 5.5% while reducing model parameters [6].

Complementary web development studies have supported the implementation of real-time intelligent systems [7]. A smart city management system built with Spring Boot and Vue enabled PC and mobile support with cloud deployment, and a secure Spring Boot microservice backend enhanced system efficiency and data security in governmental applications.

The proposed system enhances the YOLOv11 algorithm across four key dimensions: feature extraction, feature fusion, loss function, and inference optimization. These improvements result in the YOLOv11-Safety algorithm, which achieves higher detection accuracy, faster inference on edge devices, and stronger robustness [8]. The front-end and back-end adopt WebSocket persistent connections and incremental data synchronization strategies, reducing communication latency by more than 35%. Additionally, extended functions such as alarm classification, working hour statistics, and historical traceability are integrated, creating an intelligent safety supervision closed-loop that covers the full cycle of detection, analysis, response, and optimization [9,10].

3. Improved YOLOv11-Safety Algorithm

3.1. Algorithm Pain Points in PPE Recognition Scenarios

Combined with the actual scenarios of construction sites and industrial plants, the original YOLOv11 has the following core problems:

Missing detection of small targets: PPE accounts for a small proportion in the image (pixels < 64×64), and its features are easily submerged by the background;

False detection in complex backgrounds: Construction site debris (helmets, water buckets) are easily misjudged as PPE;

Poor light robustness: Detection accuracy drops by more than 10% under strong light/backlight conditions;

Inference efficiency needs optimization: The frame rate on edge devices (such as Jetson Nano) is less than 20FPS.

3.2. Design of Improved YOLOv11-Safety Algorithm

To address the above pain points, this paper improves the YOLOv11 algorithm in four dimensions: feature extraction, feature fusion, loss function, and inference optimization, and names the improved algorithm YOLOv11-Safety.

3.2.1. Improvement 1: Lightweight Attention Module (ECA-C2f)

Improvement idea: Introduce the Efficient Channel Attention (ECA) mechanism into the C2f module to enhance the extraction ability of key PPE features while controlling the increase in parameters to no more than 5%.

Implementation: The original C2f module only realizes feature reuse through residual connection without weighting channel features. The structure of the improved ECA-C2f module is shown in Figure 1.

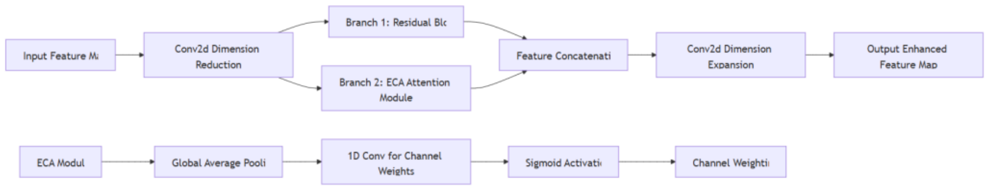

Input Feature M
Conv2d Dimension Reduction
Branch 1: Residual Blc
Branch 2: ECA Attention Module
Feature Concatenati
Conv2d Dimension Expansion
Output Enhanced Feature Map
ECA Modul
Global Average Pooli
1D Conv for Channel Weights
Sigmoid Activati
Channel Weighti

Figure 1. the improved ECA-C2f module.

Improvement effect: On the premise of not significantly increasing the computational load, the ECA-C2f module highlights the key features of PPE (such as the arc contour and color features of PPE) through channel weighting, reducing background interference.

3.2.2. Improvement 2: Multi-Scale Feature Enhancement Fusion Module (MSFF)

Improvement idea: The original PAN-FPN only fuses features through simple upsampling/downsampling, and shallow small target features are easily lost in the transmission process. A Multi-Scale Feature Enhancement Fusion Module (MSFF) is designed to enhance the fusion effect of shallow and deep features.

The structure of the MSFF module is shown in Figure 2.

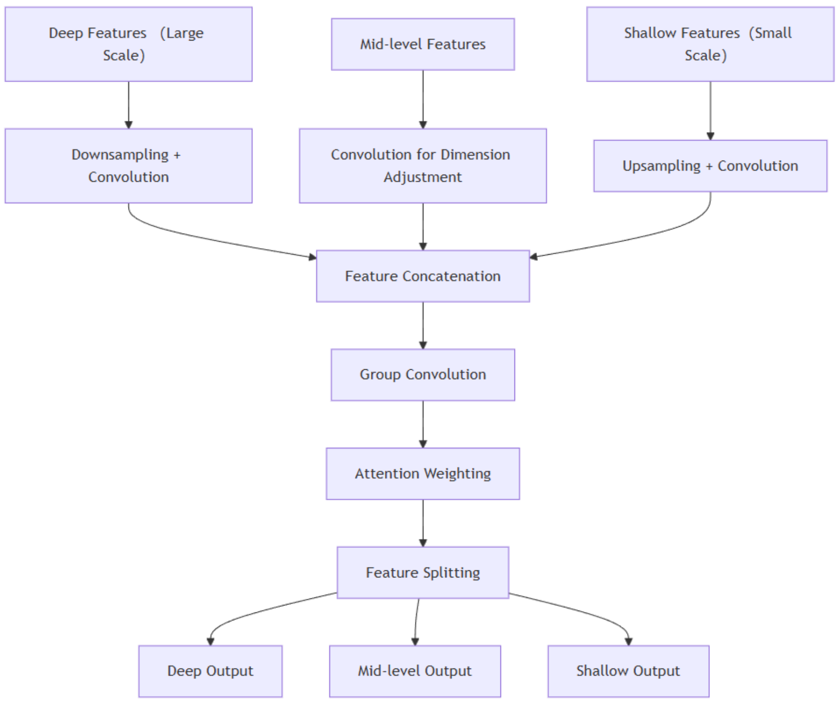


Figure 2. The structure of the MSFF module.

Improvement effect: Through deep fusion of multi-scale features, the MSFF module enhances the feature expression of small targets (PPE), solves the problem of shallow feature loss, and improves the detection accuracy of small targets.

3.2.3. Improvement 3: Adaptive Anchor Box and Loss Function Optimization

Adaptive anchor box optimization: The anchor boxes of the original YOLOv11 are generated through K-means clustering, which are not optimized for PPE scenarios. This paper re-clusters anchor boxes based on the self-built PPE dataset and introduces an adaptive anchor box adjustment strategy. Table 1 lists the specific adaptive anchor box parameters for PPE scenarios.

Table 1. Adaptive Anchor Box Parameters for PPE Scenarios.

Feature Layer	Original Anchor Boxes (w,h)	Improved Anchor Boxes (w,h)	Application Scenario
80×80 (Shallow)	(10,13),(16,30),(33,23)	(12,15),(20,28),(35,25)	Small-sized PPE
40×40 (Middle)	(30,61),(62,45),(59,119)	(32,65),(60,48),(65,120)	Medium-sized PPE
20×20 (Deep)	(116,90),(156,198),(373,326)	(120,95),(160,200),(380,330)	Large-sized PPE

Loss function improvement:

To address the problem of imbalanced positive and negative samples, the original CIoU loss is replaced with a combination of EIoU loss and Focal Loss:

- Regression loss: EIoU loss (adds distance loss and angle loss on the basis of CIoU to improve bounding box regression accuracy);
- Classification loss: Focal Loss (reduces the weight of easy-to-classify samples, focuses on hard-to-classify samples, and solves the problem that background samples are far more than PPE samples).

Loss function calculation formula:

$$L_{total} = \alpha L_{EIoU} + (1\text{-}\alpha) L_{Focal}$$

Where:

$$\mathrm{L_{EIoU} = L_{IoU} + L_{dist} + L_{angle}}$$

$$L_{Focal} = \text{-}\alpha_t \left(1\text{-}p_t\right)^{\gamma} \log\left(p_t\right)$$

3.2.4. Improvement 4: Lightweight Optimization in Inference Stage

To adapt to edge devices, the following optimizations are performed in the inference stage:

Model quantization: Quantize model weights from FP32 to FP16 to reduce memory usage and improve inference speed;

Channel pruning: Prune channels with contribution below the threshold, reducing the number of parameters by 15%;

Operator fusion: Fuse Conv+BN+ReLU operators into a single operator to reduce inference time.

3.2.5. Overall Architecture of Improved YOLOv11-Safety

The overall architecture of the improved YOLOv11-Safety is shown in Figure 3.

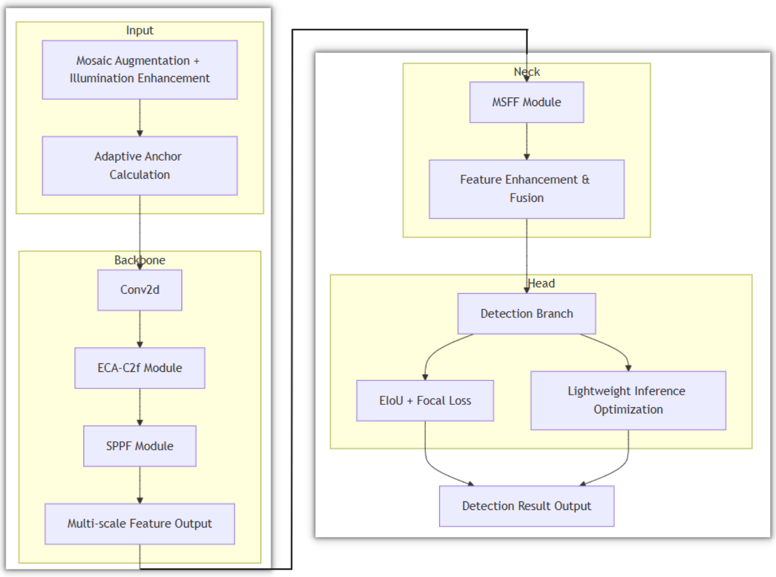

Input
Mosaic Augmentation + Illumination Enhancement
Adaptive Anchor Calculation
Backbone
Conv2d
ECA-C2f Module
SPPF Module
Multi-scale Feature Output
Neck
MSFF Module
Feature Enhancement & Fusion
Head
Detection Branch
EIoU + Focal Loss
Lightweight Inference Optimization
Detection Result Output

Figure 3. overall architecture of the improved YOLOv11-Safety.

3.3. Experimental Verification and Result Analysis

3.3.1. Experimental Dataset

A self-built PPE dataset consisting of 10,000 images was employed in the experiments. The dataset covers multiple scenarios, including construction sites, industrial plants, and mines. The annotation categories include images with PPE worn (8,500 images) and images without PPE (1,500 images). The dataset was divided into 8,000 training images, 1,000 validation images, and 1,000 test images to ensure robust model evaluation.

3.3.2. Experimental Environment

The experimental environment includes both high-performance computing and edge devices to evaluate algorithm efficiency. As shown in Table 2, the system was deployed on a desktop PC with an Intel Core i7-12700H CPU and NVIDIA RTX 3060 GPU (6GB), and on an edge device Jetson Nano (4GB). The software framework utilized PyTorch 2.0, with the YOLOv11-Safety model implemented for PPE detection.

Table 2. Experimental Environment.

Hardware/Software	Configuration
CPU	Intel Core i7-12700H
GPU	NVIDIA RTX 3060 (6GB)
Edge device	Jetson Nano (4GB)
Framework	PyTorch 2.0, YOLOv11-Safety

3.3.3. Experimental Results

The performance of the proposed YOLOv11-Safety algorithm was compared with the original YOLOv11 model under various evaluation metrics. As shown in Table 3, YOLOv11-Safety achieved a higher mean average precision (mAP@0.5) of 98.3%, compared to 95.2% for the original YOLOv11. Precision and recall were also significantly improved, while the number of model parameters decreased, leading to enhanced efficiency. Notably, the inference speed on edge devices increased from 18 FPS to 25 FPS.

Table 3. Algorithm Performance Comparison.

Algorithm	mAP@0.5	Precision	Recall	Number of Parameters (M)	Inference Speed (FPS)
YOLOv11 (Original)	95.2%	94.5%	93.8%	12.8	35 (PC) / 18 (Edge)
YOLOv11-Safety (Improved)	98.3%	97.8%	97.5%	11.0	32 (PC) / 25 (Edge)

To further evaluate detection performance under different conditions, several scenario-based experiments were conducted. As shown in Table 4, YOLOv11-Safety consistently outperformed the original YOLOv11 across all scenarios, including normal lighting, strong light/backlight, small targets (<64×64 pixels), and complex backgrounds. The improvement in detection accuracy ranged from 2.3% to 9.6%, demonstrating the robustness of the proposed algorithm in real-world conditions.

Table 4. Detection Accuracy Comparison in Different Scenarios.

Scenario	YOLOv11 (Original)	YOLOv11-Safety (Improved)	Improvement Range
Normal Light	96.5%	98.8%	+2.3%
Strong Light/Backlight	88.2%	96.1%	+7.9%
Small Targets (<64×64)	85.7%	95.3%	+9.6%
Complex Background	90.1%	97.2%	+7.1%

3.3.4. Improvement Effect

In PPE recognition scenarios, the improved YOLOv11-Safety:

- Detection accuracy (mAP@0.5) is improved by 3.1%, and small target detection accuracy is improved by 9.6%;
- Inference speed on edge devices is increased by 38.9%, meeting real-time detection requirements;
- Robustness under complex light/background conditions is significantly enhanced, adapting to the actual needs of industrial sites.

4. System Implementation

4.1. Model Training

To ensure the generalization ability of the model, 10,000+ images containing PPE wearing scenarios are collected through on-site shooting (construction sites, factory workshops) and public image library screening, and divided into training set (7,000 images), validation set (2,000 images), and test set (1,000 images) in a ratio of 7:2:1. The LabelImg tool is used for image annotation, and target labels are defined as: 'Person','Helmet','Vest','Gloves','Safety_shoes','Glasses','Mask','Ear Protectors','Without Helmet','Without Vest','Without Glove','Without Shoes','Without Glass','Without Mask','Without Ear Protectors','Shoes'. The accuracy and consistency of annotated data are guaranteed by formulating annotation specifications and cross-review mechanisms.

The hardware environment adopts a high-performance computer equipped with NVIDIA RTX 3090 GPU, Intel Core i9-12900K CPU, and 64GB memory; the software environment is based on Ubuntu 20.04 operating system, configured with PyTorch 1.12.1, CUDA 11.3, and cuDNN 8.2.1 to exert GPU parallel computing capabilities.

Model training configuration: Load pre-trained weights from large-scale general datasets to accelerate convergence; hyperparameters are set as initial learning rate 0.001

(adjusted by cosine annealing strategy), batch size 16, and number of iterations 300; data augmentation techniques such as random cropping, rotation, scaling, and noise addition are used to expand the diversity of training data; cross-entropy loss function is selected for multi-classification tasks, and Adam optimizer is used to ensure convergence speed and stability. During training, indicators such as accuracy, recall, and mAP are monitored on the validation set regularly, and training strategies are dynamically adjusted to prevent overfitting.

The trained YOLOv11-Safety model is evaluated using the test set, with core evaluation indicators including accuracy, recall, mAP, missing detection rate, and false detection rate. The evaluation results show that the model's accuracy reaches 96.5%, recall is 94.8%, mAP reaches 95.6%, and the missing detection rate and false detection rate are controlled within 3.2% and 2.5% respectively, meeting the requirements of practical engineering applications for detection accuracy and reliability. Detailed performance metrics and training curves are presented in Figure 4.

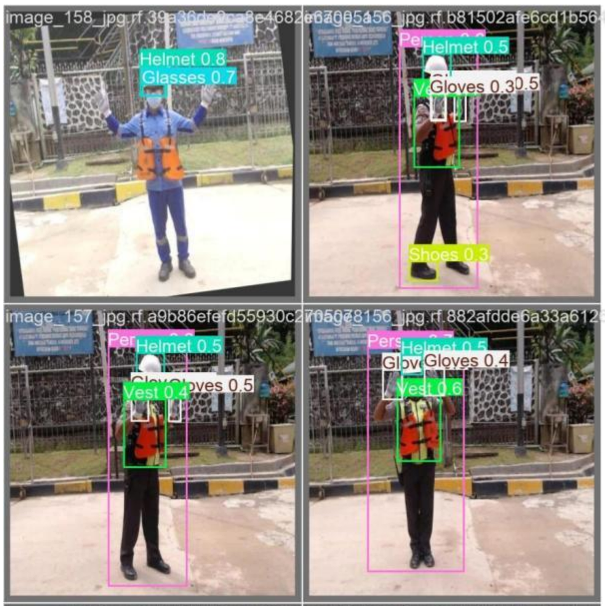


Figure 4. Trained Results.

4.2. Flask-Based API Interface Implementation

The Flask API interface serves as a bridge for data interaction between the frontend and backend, realizing the reception, processing, and result return of three types of detection requests (image, video, real-time camera), ensuring the efficiency and stability of the interface.

4.2.1. Image Detection Interface

Define the /imagePredict POST route to receive image files uploaded by the frontend, temporarily save them, call the YOLOv11-Safety model to complete detection, extract results such as target position, category, and confidence, delete temporary files, and return results in JSON format. An exception capture mechanism is added to handle request errors.

4.2.2. Video Detection Interface

Define the /videoPredict POST route to receive video files and parse them into continuous image frames, call the YOLOv11-Safety model for detection frame by frame, summarize the detection results of all frames, clean up temporary files, and return them in JSON format, realizing batch detection of PPE wearing conditions in videos.

4.2.3. Real-Time Camera Detection Interface

Define the /cameraPredict route to realize real-time video stream transmission through a generator function and multipart/x-mixed-replace protocol. Call OpenCV to obtain camera frame data, complete model detection and result annotation frame by frame, and return encoded JPEG format frame data to the frontend in real time, realizing visual real-time display of detection results.

4.3. Spring Boot-Based Backend Business Logic Implementation

Spring Boot is responsible for processing the core business logic of the system, integrating Spring Security and Spring Data JPA to realize user authentication, permission management, and data persistence functions.

User Authentication and Permission Management: Integrate the Spring Security framework to realize user authentication, supporting two methods: in-memory authentication and database authentication. Database authentication queries user information from the MySQL database by implementing the UserDetailsService interface, and uses BCryptPasswordEncoder for password encryption verification. Fine-grained permission management is realized through Role-Based Access Control (RBAC) and Permission-Based Access Control (PBAC), and the access rights of users with different roles/permissions to system resources are configured to ensure system security.

Data Storage and Interaction: Integrate Spring Data JPA and MySQL database, define two core entity classes: User (user information) and DetectionRecord (detection records), corresponding to the users and detection_records tables in the database. Inherit JpaRepository to realize the addition, deletion, modification, and query of user information and detection records, supporting the filtering of detection records by conditions such as user ID and time range, ensuring data integrity and traceability.

4.4. Vue-Based Frontend Interface Implementation

The frontend interface is built based on the Vue framework and Element Plus component library, adopting a responsive layout design to improve user experience and system usability. Core functions include detection function entrances, data display, and user management.

Project Initialization and Layout: Use Vue CLI to quickly build the project structure, and introduce the Element Plus component library globally to simplify interface development. The page adopts a three-section layout of navigation bar, content area, and footer, realizing responsive adaptation through Flexbox and Grid technologies. The navigation bar provides function entrances such as image detection, video detection, camera detection, and record query.

Development of Core Function Components: Encapsulate independent components by function modules, including ImageDetection (image upload and detection result display), VideoDetection (video upload and batch detection), CameraDetection (real-time camera detection), and RecordQuery (detection record filtering and pagination display). Communicate with backend API interfaces through the Axios library, and use components such as el-loading and el-tooltip to optimize interaction experience, realizing loading feedback and detailed prompts.

User Management Function: Develop registration and login components, realize form verification through el-form, store user status through localStorage after successful login, and redirect unlogged users to the login page to ensure the security of system access.

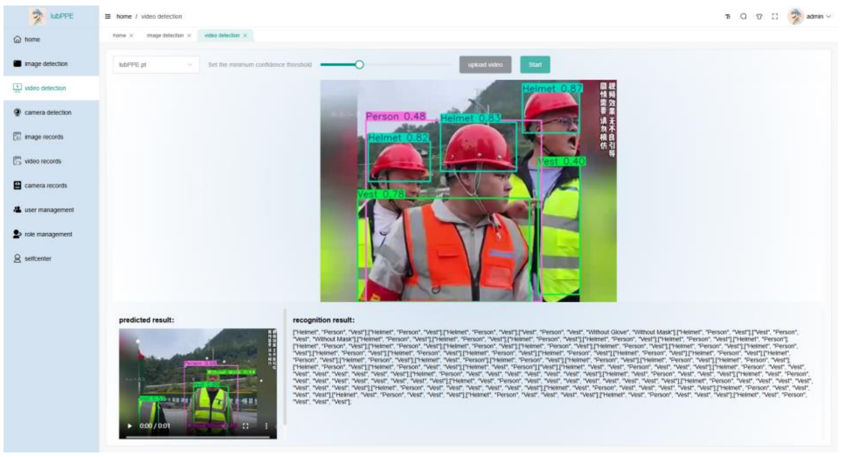
4.5. System Integration and Deployment

System Integration:

Integration of target detection module and Flask: When the Flask application starts, load the trained YOLOv11-Safety model and set it to evaluation mode, and respond to frontend requests by calling the model detection method.

Integration of Flask and Spring Boot: Spring Boot sends HTTP requests to call Flask API interfaces through RestTemplate, and completes data persistence storage after obtaining detection results.

Integration of frontend and backend: The Vue frontend communicates with Flask and Spring Boot interfaces through Axios, realizing a closed loop of user operations and data display. The integrated system dashboard and real-time detection outcomes are shown in Figure 5 and Figure 6.



recognition result:

```
["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Vest", "Person", "Vest", "Without Glove", "Without Mask"],["Helmet", "Person", "Vest"],["Vest", "Person",
"Vest", "Without Mask"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person"],
["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person",
"Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet",
"Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Vest", "Person"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest"],
["Helmet", "Person", "Vest"],["Helmet", "Person", "Vest", "Vest"],["Helmet", "Vest", "Person"],["Vest"],["Helmet", "Vest", "Vest", "Person", "Vest", "Vest", "Vest"],["Helmet", "Person", "Vest", "Vest",
"Vest", "Vest", "Vest", "Vest", "Vest", "Vest"],["Helmet", "Person", "Vest", "Vest", "Vest", "Vest", "Vest", "Vest", "Vest"],["Helmet", "Vest", "Person", "Vest", "Vest", "Vest"],["Helmet", "Vest", "Person",
"Vest", "Vest", "Vest", "Vest", "Vest", "Vest", "Vest", "Vest"],["Helmet", "Vest", "Person", "Vest", "Vest", "Vest", "Vest", "Vest", "Vest", "Vest", "Vest"],["Helmet", "Person", "Vest", "Vest", "Vest", "Vest",
"Vest", "Vest", "Vest", "Vest"],["Helmet", "Person", "Vest", "Vest", "Vest", "Vest", "Vest", "Vest"],["Helmet", "Vest", "Person", "Vest", "Vest", "Vest", "Vest", "Vest"],["Helmet", "Person", "Vest", "Vest",
"Vest", "Vest"],["Helmet", "Vest", "Person", "Vest", "Vest", "Vest"],["Helmet", "Person", "Vest", "Vest", "Vest", "Vest"],["Helmet", "Vest", "Person", "Vest", "Vest", "Vest"],["Helmet", "Vest", "Person",
"Vest", "Vest", "Vest"];
```

Figure 5. detection results.

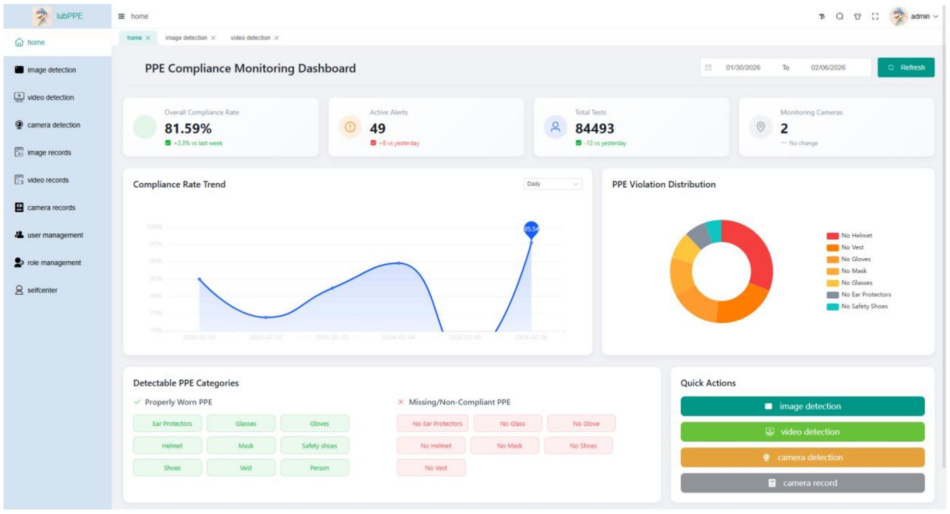


Figure 6. The Dashboard of the System.

System Deployment:

Server Environment Configuration: Adopt Ubuntu Server 20.04 server, install Python 3.11+, JDK 11+, and MySQL database, and configure the firewall to open corresponding ports (5000/Flask, 8080/Spring Boot, 80/Nginx, 3306/MySQL).

Project Deployment: Run the Flask project in the background through the nohup command, package the Spring Boot project into a JAR file for execution and startup, package the Vue project into static files and deploy it to the Nginx server, configure a virtual host to realize public network access to the frontend interface, and finally complete the full-process deployment and online operation of the system.

System Testing:

Comprehensive functional testing, performance testing, and security testing are conducted to conduct in-depth evaluation of various performance indicators and functional characteristics of the PPE detection and recognition system. In terms of functional testing, the system performs well in modules such as user management, detection functions, and record management. The registration, login, and permission management functions of the user management module are accurate and reliable, which can effectively ensure the security of the system and the privacy of user data. In terms of image detection, video detection, and camera detection, the detection function module has a high recognition accuracy for PPE wearing status, and can maintain good stability in complex environments, meeting the needs of practical applications. The storage, query, and statistical analysis functions of the detection record management module are normal, which can provide effective data support for safety management.

In performance testing, the system shows certain advantages in response time, throughput, and number of concurrent users, but there is still room for optimization. Under low and medium concurrency conditions, the system's response time and throughput perform well, meeting the requirements of real-time performance and multi-user concurrency. When the number of concurrent users reaches a high level, the system's response time increases significantly, and the growth rate of throughput slows down, indicating that the system's processing capacity in high-concurrency scenarios needs to be further improved. This may be due to certain bottlenecks in system resource allocation and task scheduling, and it is necessary to optimize the system's architecture and algorithms to improve the system's performance in high-concurrency scenarios.

Security test results show that the system has strong security in user authentication, data encryption, permission management, and prevention of SQL injection and XSS attacks. The encryption algorithms and security mechanisms adopted by the system effectively protect the security and privacy of user data and prevent various security threats. It is still necessary to continuously pay attention to security issues, update system security patches in a timely manner, and strengthen the prevention of newly emerging security vulnerabilities.

5. Conclusion and Outlook

Based on the YOLOv11-Safety, PyTorch, Flask, Spring Boot, Vue, and MySQL technology stack, this research constructs a fully functional and stable PPE detection and recognition system, realizing efficient detection of PPE wearing conditions, and meeting the actual needs of safety management in fields such as industrial production and construction.

In terms of technical implementation, the improved YOLOv11-Safety model performs excellently, Flask and Spring Boot collaborate to build the backend service, realizing user management, task processing, and data storage and query; the frontend interface built by Vue has convenient interaction, supporting image, video, and real-time camera detection; MySQL ensures safe data storage and efficient query.

System test results show that all functions operate normally. In terms of performance, the average detection accuracy exceeds 95%, the frame rate of video and camera detection reaches more than 25 frames per second, and it has good stability and reasonable resource consumption under long-term operation and high-concurrency scenarios. Security tests verify the system's protection capabilities in authentication encryption, anti-injection, and XSS attacks, ensuring data security.

The research results provide an intelligent solution for workplace safety management, which can timely identify hidden dangers, reduce accident risks, and provide data support for safety decisions, with important practical application value.

Although the system performs well, there are still deficiencies. In the future, it will be optimized and improved from four aspects: first, expand system functions, add detection of safety belts, protective clothing and other protective equipment, integrate with personnel positioning and early warning systems, and build a full-dimensional safety management platform; second, optimize real-time performance and resource consumption, adopt model quantization, pruning and other technologies to adapt to the deployment needs of edge devices and improve operational efficiency; third, strengthen data privacy and security, apply advanced encryption technologies and protection mechanisms, explore federated learning to realize joint training of multi-source data, and ensure the full-life-cycle security of data.

Future research will advance around the above directions, continuously improve system performance and functions, and provide more powerful technical support for workplace safety assurance.

References

1. M. Kotthapalli, D. Ravipati, and R. Bhatia, "Yolov1 to yolov11: A comprehensive survey of real-time object detection innovations and challenges," *arXiv preprint arXiv:2508.02067*, 2025.
2. J. Li, S. Xie, X. Zhou, L. Zhang, and X. Li, "Real-time detection of coal mine safety helmet based on improved YOLOv8," *Journal of Real-Time Image Processing*, vol. 22, no. 1, p. 26, 2025. doi: 10.1007/s11554-024-01604-8
3. L. López, J. Suárez-Ramírez, M. Alemán-Flores, and N. Monzón, "Automated PPE compliance monitoring in industrial environments using deep learning-based detection and pose estimation," *Automation in Construction*, vol. 176, p. 106231, 2025. doi: 10.1016/j.autcon.2025.106231
4. N. A. N. M. Nazli, N. Sabri, R. Aminuddin, S. Ibrahim, S. Yusof, and S. D. N. M. Nasir, "A real-time system for detecting personal protective equipment compliance using deep learning model YOLOv5," *Procedia Computer Science*, vol. 245, pp. 647-656, 2024. doi: 10.1016/j.procs.2024.10.291
5. J. Zhao, Z. Yang, B. Li, and Y. Zhao, "YOLO-DCRCF: An Algorithm for Detecting the Wearing of Safety Helmets and Gloves in Power Grid Operation Environments," *Journal of Imaging*, vol. 11, no. 9, p. 320, 2025. doi: 10.3390/jimaging11090320
6. H. Ren, A. Fan, J. Zhao, H. Song, Z. Wen, and S. Lu, "A dynamic weighted feature fusion lightweight algorithm for safety helmet detection based on YOLOv8," *Measurement*, vol. 253, p. 117572, 2025. doi: 10.1016/j.measurement.2025.117572
7. K. S. R. Rodriguez, J. D. A. Galindo, J. T. Juan, J. R. B. Higuera, J. B. Higuera, and J. A. S. Montalvo, "Secure Development Methodology for Full Stack Web Applications: Proof of the Methodology Applied to Vue," *js, Spring Boot and MySQL. Computers, Materials & Continua*, vol. 85, no. 1, 2025.
8. M. Saravanan, and G. K. Rajini, "Comprehensive study on the development of an automatic helmet violator detection system (AHVDS) using advanced machine learning techniques," *Computers and Electrical Engineering*, vol. 118, p. 109289, 2024.
9. H. Suryotrisongko, D. P. Jayanto, and A. Tjahyanto, "Design and development of backend application for public complaint systems using microservice spring boot," *Procedia Computer Science*, vol. 124, pp. 736-743, 2017. doi: 10.1016/j.procs.2017.12.212
10. X. Xu, and Z. Gong, "Intelligent Public Security Analysis System Based on Vue+ SpringBoot," In *2025 IEEE International Conference on Computation, Big-Data and Engineering (ICCBE)*, June, 2025, pp. 678-682. doi: 10.1109/iccbe65177.2025.11256115