Jurnal Masyarakat Informatika, 16(1), 2025
DOI:10.14710/jmasif.16.1.71622
ISSN: 2777-0648

Development and Optimization of a Construction Personal Protective Equipment (PPE) Detection Model on YOLOv8 Architecture

Zidan Rafindra Utomo[*1)], Prajanto Wahyu Adi[1)], Priyo Sidik Sasongko[1)], Gohar Rahman[2)]
[1)]*Departemen Informatika, Universitas Diponegoro, Kota Semarang, Indonesia*
[2)] *Faculty of Computing and Informatics, University Malaysia Sabah, Kinabalu, Malaysia*
[*] *Corresponding author: zidanutomo@students.undip.ac.id*

Abstract
Workplace safety in the construction industry continues to face challenges due to frequent accidents linked to non-compliance with Personal Protective Equipment (PPE) regulations. Manual supervision is limited by human error and inefficiency, necessitating automated detection systems. However, prior models such as YOLOv5 demonstrate limited performance in detecting underrepresented PPE violations, particularly "no-helmet" and "no-vest" instances, due to class imbalance in the dataset. This study addresses these limitations by optimizing the YOLOv8 model through grid search-based hyperparameter tuning and enhancing generalization via Mosaic and MixUp data augmentation techniques. Trained on the imbalanced Construction Safety subset of Roboflow-100, the improved YOLOv8 model achieves a mAP@0.50 of 0.921 and mAP@50-95 of 0.533, surpassing earlier configurations. These findings demonstrate significant gains in detection accuracy—especially for rare PPE violations—supporting the development of more robust and scalable solutions for real-time workplace safety monitoring.

Keywords : *Workplace safety, PPE detection, YOLOv8, hyperparameter tuning, data augmentation*

1 Introduction

Workplace safety is a critical issue that must be addressed in the construction industry, as the number of work-related accidents each year is on the rise. According to data from the Ministry of Manpower of the Republic of Indonesia, there is a growing trend in workplace incidents that are the result of noncompliance with the use of Personal Protective Equipment (PPE). Material damage and human casualties are substantial losses as a consequence of accidents in the construction industry. The use of personal protective equipment (PPE) is mandatory for all construction workers in order to reduce the risk of accidents, as specified in Occupational Safety and Health (OSH) regulations in Act of Republic of Indonesia No. 1 of 1970 regarding safety [1].

According to the Occupational Safety and Health Administration (OSHA), proper use of personal protective equipment can minimize workplace accidents by up to 40%. However, PPE manual supervision compliance remains a concern due to a limited number of safety inspectors and the possibility of human error [2]. Manual monitoring is also time-consuming and may not be feasible in dynamic or high-risk environments where rapid detection of non-compliance is crucial. As a result, an automated solution to PPE compliance monitoring is required [3], offering the potential for real-time, consistent, and scalable enforcement across various industrial settings.

Received: 14 March 2025; Revised: 11 April 2025; Accepted: 28 April 2025; Published: 31 May 2025

Several studies have shown that artificial intelligence (AI)-based technology can increase PPE compliance by automating detection [4]. YOLOv8, a deep learning-based object detection model, has demonstrated superior performance over previous methods. Compared to Faster R-CNN and YOLOv5, YOLOv8 is faster and more accurate [5]. Previous research has investigated PPE detection using various methods. According to Barlybayev et al. [6], YOLOv8 detected helmets and safety vests with a mean Average Precision (mAP) of 92.9%. Çiftçi et al. [7] found that YOLOv8 outperformed YOLOv5 by 6% in accuracy. Furthermore, hyperparameter optimization utilizing the grid search method has been shown to improve model performance, as demonstrated by Popek et al. [8].

Further research has looked into the effectiveness of several PPE detection technologies. A study [6] examined several YOLOv8 model sizes and discovered that YOLOv8x (extra-large) had the best performance, with a mAP of 0.929. In comparison, [9] used the R-CNN method to achieve a higher mAP of 96% in PPE detection while incurring a much larger computational cost than YOLOv8. Another study [5] discovered that YOLOv8 outperformed R-CNN for object detection tasks in aquatic environments. Detection speed is also important in automated PPE monitoring systems. A study by [10] tested YOLOv5 and Faster R-CNN for space object recognition and discovered that while Faster R-CNN had a higher mAP, YOLOv5 was up to ten times faster. This emphasizes the necessity of detecting speed, especially in real-time application in building projects. Furthermore, research by [7] examined YOLOv5 and YOLOv8 on the same dataset and found a 5.3% accuracy improvement with YOLOv8, supporting the advancements of the most recent YOLO model over its predecessors.

Although earlier approaches such as Faster R-CNN and YOLOv5 have contributed significantly to automated PPE detection, there remains a substantial gap in balancing detection accuracy and computational efficiency, an area where YOLOv8 excels. While Faster R-CNN can achieve high mAP scores, it incurs high inference times, making it impractical for real-time deployment in dynamic environments like construction sites [9], [10]. YOLOv5, on the other hand, offers faster detection but at the cost of reduced accuracy, particularly for underrepresented PPE classes. In contrast, YOLOv8 has demonstrated superior performance by achieving a mean Average Precision (mAP) of 92.9% in detecting helmets and vests [6], and has consistently outperformed YOLOv5 by 5-6% in accuracy across multiple studies [7]. Additionally, YOLOv8 maintains low computational overhead, allowing it to outperform R-CNN variants even in complex detection scenarios such as underwater environments [5]. As further evidenced by hyperparameter optimization studies [8], the integration of tuning and augmentation techniques in YOLOv8 yields significant performance gains, reinforcing its advantage as a robust, efficient, and scalable solution for real-time PPE compliance monitoring.

This study uses the grid search approach for hyperparameter optimization to improve detection performance. In order to get an ideal model configuration, grid search makes it possible to investigate the best parameter combinations, such as learning rate, batch size, and optimizer [11]. The aim is to maximize detection accuracy, improving PPE monitoring in construction sites. Additionally, data augmentation is employed to address the problem of class imbalance, which can hinder model generalization [12]. In PPE datasets, underrepresented classes negatively affect detection accuracy. According to a study [13], augmentation techniques such as Mosaic and MixUp can effectively address this issue by increasing data diversity and improving model robustness. These methods are especially effective in one-stage detectors like YOLOv5, where traditional sampling and loss-reweighting methods yield limited gains. Integrating these augmentations into YOLOv8 training further improves mAP by generating more representative and diverse training samples [13].

2 Research Methods

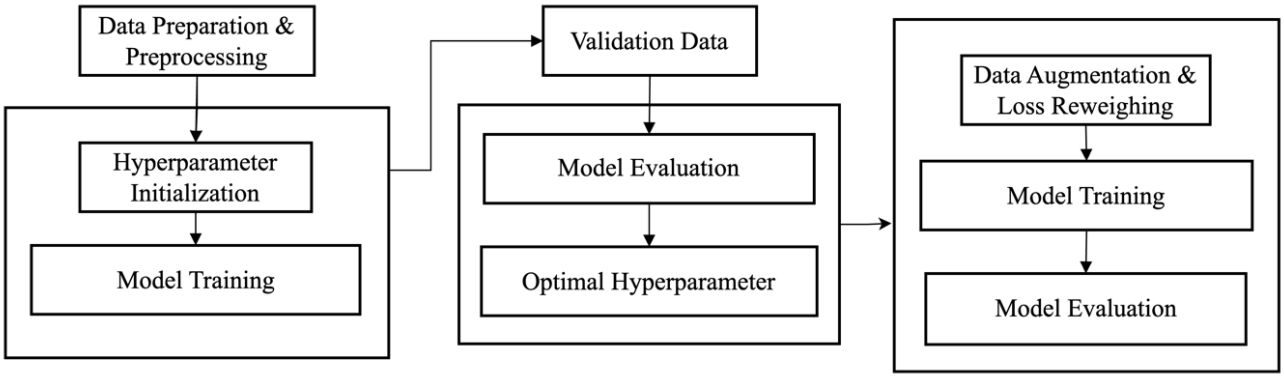

Figure 1 Research Method

This study adopts a research methodology composed of four main phases: (1) Data Preparation, (2) Model Development and Hyperparameter Optimization, (3) Data Augmentation for Class Imbalance, and (4) Model Evaluation. Each phase is designed to incrementally enhance the PPE detection performance using YOLOv8, guided by a structured workflow as illustrated in Figure 1. The first stage, Data Preparation, includes data collection and preliminary preprocessing. The Model Development phase then includes hyperparameter initialization, YOLOv8 model training, and grid search to optimize hyperparameter selection. The grid search strategy was selected for hyperparameter tuning due to its systematic and exhaustive nature, allowing the exploration of all possible combinations of selected training parameters to identify the optimal configuration for detection accuracy and convergence stability [11]. After determining the optimal hyperparameters, data augmentation is used to address class imbalances, particularly for No-Helmet and No-Vest. Mosaic and MixUp are augmentation strategies that improve feature diversity and detection performance in imbalanced datasets [13]. The Model Validation phase assesses performance on the validation dataset by calculating mean Average Precision (mAP) from precision, and recall [14].

2.1 Data Preparation

The dataset used in this study is Construction Safety, which is a component of Roboflow-100. RF100 is a collection of 100 datasets from seven different image domains, including satellite imagery, microscopy, and gaming, totaling 224,714 images and 805 class labels [15]. The goal of RF100 is to provide a semantic and multi-domain benchmark for assessing model generalization using real-world data. The Construction Safety dataset used in this study is a subset of RF100 that focuses on Personal Protective Equipment (PPE) detection on construction sites. This dataset includes a variety of PPE classes, such as helmets and vests, photographed under various lighting conditions and perspectives.

Figure 2 Construction Safety Dataset

Figure 2 shows sample images from the Construction Safety dataset. Each bounding box label contains five distinct classes: helmet, no-helmet, vest, no-vest, and person. The Construction Safety dataset is divided into three sections: training, validation, and testing, with 80% for training, 10% for validation, and 10% for testing. This structured partitioning ensures that the model has enough data for training while also including a validation set for periodic performance monitoring and a test set for final accuracy evaluation on unseen data. The training set consists of 2,116 instances of helmets, 94 instances of no-helmet, 741 instances of no-vest, 2,362 instances of persons, and 1,073 instances of vests. The validation set includes 232 instances of helmets, 11 instances of no-helmet, 90 instances of no-vest, 241 instances of persons, and 141 instances of vests. The testing set contains 195 instances of helmets, 24 instances of no-helmet, 61 instances of no-vest, 214 instances of persons, and 129 instances of vests.

Table 1 Class Distribution of Dataset

	Classes	Counts
1	Helmet	2.543
2	No-Helmet	129
3	No-Vest	892
4	Person	2.817
5	Vest	1.343
Total	-	7.724

Table 1 presents the distribution of classes in the Construction Safety dataset. Despite the structured partitioning, the dataset still shows class imbalance, particularly in the No-Helmet and No-Vest categories, which are significantly underrepresented when compared to other PPE classes. The No-Helmet class has only 94 training instances, which is 22 times fewer than the most frequent class (Person, with 2,362 instances). Similarly, the No-Vest class has 741 instances, which is disproportionately low when compared to the other categories. This imbalance presents a challenge for model training because the model may develop a bias toward majority classes, resulting in poor detection performance for underrepresented PPE violations. To address this issue, data augmentation techniques such as Mosaic and MixUp are used to artificially increase the number of No-Helmet and

No-Vest samples, thereby increasing model robustness and improving detection accuracy for critical safety violations.

Data preprocessing standardizes image dimensions for PPE detection model training, resizing all images to 640×640×3 pixels. This resizing process keeps inputs consistent, allowing the model to process data efficiently and make accurate predictions. Resize and padding operations are used to maintain the original aspect ratio while avoiding distortion [16].

2.2 Model Development and Hyperparameter Optimization

This stage focuses on building and optimizing the YOLOv8 model for PPE detection. The grid search approach is used to evaluate combinations of optimizers, batch sizes, and learning rates. Each configuration is trained and validated to measure its impact on model accuracy. The best-performing setup is then selected for further enhancement through data augmentation.

Table 2 Hyperparameter Range for Grid Search

Parameter	Value
Optimizer	['Adam', 'RMSProp', 'SGD']
Batch Size	[8, 16]
Learning Rate	[0,01; 0,001; 0,0001]

The YOLOv8 model's hyperparameters are selected using a grid search approach to identify the optimal configuration based on variations in key parameters [11]: optimizer, batch size and learning rate. The study investigates 18 hyperparameter combinations, including three optimizer types (Adam, RMSProp, and SGD), two batch sizes (8 and 16), and three learning rates (0.01, 0.001, and 0.0001). Each configuration is trained for 100 epochs, as described in Table 2.

The choice of optimizer plays a central role in determining the convergence behavior and overall training efficiency. Adam was included for its capability to combine momentum and adaptive learning rate adjustments, which accelerates convergence even on sparse or noisy data [11]. RMSProp is selected for its strength in smoothing out gradient updates on non-stationary loss surfaces, often found in object detection tasks [17]. SGD, though simpler and slower to converge, serves as a foundational baseline and is useful to assess the relative performance gains offered by adaptive optimizers [17]. These optimizers represent diverse strategies in handling gradient-based updates, allowing for robust comparative evaluation.

Additional important hyperparameters are batch size and learning rate, which significantly influence the performance of model training. Smaller batch sizes can introduce noise in gradient updates, which may enhance generalization. In contrast, larger batch sizes stabilize training but may necessitate adjustments in the learning rate to avoid underfitting or convergence issues [17]. Learning rates require careful calibration; if set too high, the model risks divergence, while if set too low, convergence may occur at an unacceptably slow pace. Optimal performance of training frequently results from the independent tuning of the learning rate for each optimizer, rather than utilizing a shared search space [17]. This underscores the significance of tuning protocols as a critical factor in achieving successful training outcomes.

Model training is carried out with the Ultralytics YOLO library, which offers a simplified programming interface for training and validation. To ensure consistency during initial model learning,

the training process is carried out without the use of data augmentation. Each hyperparameter configuration is iteratively trained on the training dataset, with parameters like optimizer, batch size, and learning rate influencing the learning process and weight updates to reduce loss.

During training, the model's performance is assessed at each epoch using recall, precision, and mean Average Precision (mAP) at various thresholds (mAP50-95 and mAP50) to determine its accuracy in PPE detection. The validation dataset is used to perform single-fold validation. The model checkpoints are saved based on the highest mAP score achieved at each epoch, as well as the final epoch. This ensures that the best-performing model from each hyperparameter configuration is saved for later analysis in order to select the optimal hyperparameter set.

Table 3 Grid Search Hyperparameter Configurations for Each Model

Model ID	Optimizer	BatchSize	LearningRate
model_001	Adam	8	0,01
model_002	Adam	8	0,001
model_003	Adam	8	0,0001
model_004	Adam	16	0,01
model_005	Adam	16	0,001
model_006	Adam	16	0,0001
model_007	RMSProp	8	0,01
model_008	RMSProp	8	0,001
model_009	RMSProp	8	0,0001
model_010	RMSProp	16	0,01
model_011	RMSProp	16	0,001
model_012	RMSProp	16	0,0001
model_013	SGD	8	0,01
model_014	SGD	8	0,001
model_015	SGD	8	0,0001
model_016	SGD	16	0,01
model_017	SGD	16	0,001
model_018	SGD	16	0,0001

Each trained model is assigned a unique identifier, as shown in Table 3, to allow for performance tracking and comparison across different configurations. During training, the model's performance is assessed at each epoch using recall, precision, and mean Average Precision (mAP) at various thresholds (mAP50-95 and mAP50) to determine its accuracy in PPE detection. The validation dataset is used to perform single-fold validation. The model checkpoints are saved based on the highest mAP score achieved at each epoch, as well as the final epoch. This ensures that the best-performing model from each hyperparameter configuration is saved for later analysis in order to select the optimal hyperparameter set. Each trained model is assigned a unique identifier, as shown in Table 3, to allow for performance tracking and comparison across different configurations.

2.3 Data Augmentation

Following hyperparameter optimization, data augmentation techniques are used to address class imbalances in the dataset. The No-Helmet and No-Vest classes are significantly underrepresented in the training set, with only 94 and 741 instances, respectively, as opposed to the dominant classes, Person (2,362 instances) and Helmet (2,116 instances). This imbalance can result in biased model learning, in which the detector predicts frequent classes but fails to detect rare PPE violations.

To address this, post-training augmentation is applied using Mosaic and MixUp techniques, which improve model generalization by increasing sample diversity and enhancing feature learning [13]. This augmentation method improves the detection accuracy of PPE violations in real-world construction settings by ensuring more balanced class representation.

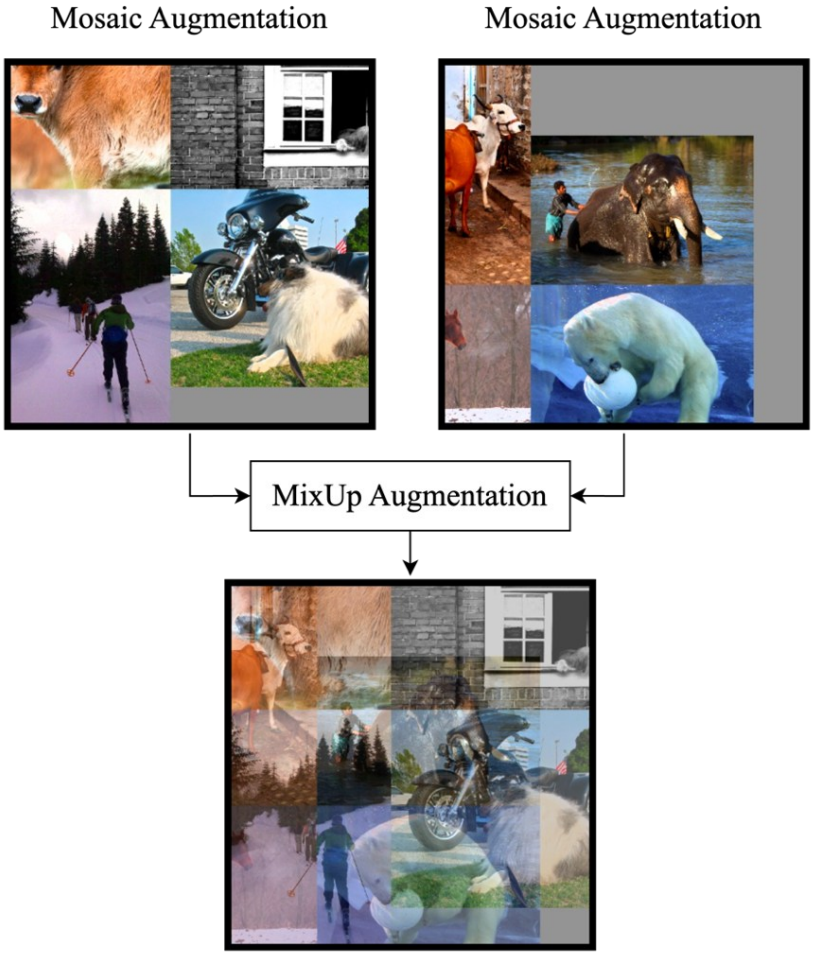

Figure 3 Mosaic and Mixup Augmentation Visualization [13]

Mosaic augmentation combines four randomly selected training images into a single composite image by partitioning the input space and stitching image segments together, resulting in a greater variation in object scales, spatial distributions, and contextual backgrounds. This augmentation broadens the range of object placements within an image, allowing the model to better learn feature representations across different scenarios [18]. Following mosaic augmentation, Mixup augmentation is used to create new training samples by blending the mosaic-generated image with a randomly selected image. This is accomplished through linear interpolation of pixel values and bounding box labels, with a randomly chosen mixing coefficient determining each image's contribution to the final

augmented sample. Mixup reduces overfitting by introducing smooth transitions between different object instances, encouraging the model to learn more general decision boundaries [13]. Figure 3 depicts how this process works.

2.4 Model Evaluation

The model evaluation process is conducted to assess the final performance of the selected YOLOv8 model on the independent testing dataset. The key metric used in this evaluation is mean Average Precision (mAP), which is derived from the relationship between precision and recall at various Intersection over Union (IoU) thresholds [14]. Precision measures the proportion of correct positive detections among all predicted positives, while recall quantifies the proportion of actual positives correctly identified by the model. Together, these metrics form the basis of the Precision-Recall (PR) curve, where the AP is computed as the area under the curve for each class.

The overall performance of the model is measured using mean Average Precision (mAP), calculated as the average of the AP values across multiple object classes and IoU thresholds. For example, mAP50 evaluates performance at a fixed IoU threshold of 0.5, while mAP50-95 represents the average of AP values computed at thresholds from 0.5 to 0.95 in 0.05 increments. These metrics provide a comprehensive assessment of the model's accuracy and robustness in object detection. A higher mAP50 indicates strong sensitivity, while mAP50-95 captures precision under stricter localization requirements. This evaluation confirms whether the optimized YOLOv8 model achieves the desired level of generalization and detection performance on unseen data [14].

3 Results and Discussion

The research was conducted exclusively in a Jupyter Notebook environment on a local server. The specifications of the local computing device used in this research are presented in Table 4. The dataset was stored on local storage for fast access and responsiveness on training and data processing. However, the use of a local GPU imposes limitations, particularly in handling large batch sizes and training configurations involving high-resolution inputs or extended epochs, as the limited video memory constrains memory-intensive operations.

Table 4 Research Device Specifications

Component	Specification
Processor	AMD Ryzen 5 5600X
RAM	32GB
GPU	NVIDIA RTX 4060, 8GB
CUDA Version	12.01
GPU Driver Version	566.03.00
PyTorch Version	2.4.1+cu121
Ultralytics Version	8.3.95
Python Version	3.10.11

The training results of the YOLOv8 model using various hyperparameter combinations obtained through a grid search process [19] are presented in Table 5. The hyperparameter combinations consist of three types of optimizers (Adam, RMSProp, and SGD), two batch size values (8 and 16), and three learning rate values (0.01, 0.001, and 0.0001). Each combination was tested to assess model performance using key metrics like mean Average Precision (mAP) at various Intersection over Union (IoU) thresholds and object classes (helmet, no-helmet, no-vest, person, and vest). A total of 18 hyperparameter combinations were tested, and each model was assigned a unique ID based on the training sequence.

Table 5 Grid Search Results

ID	*Optimizer*	*BatchSize*	*LearningRate*	*mAP50-95*	*mAP50*	*mAP75*	*recall*
model_001	Adam	8	0.01	0.5037	0.8671	0.52	0.86226
model_002			0.001	0.4972	0.8922	0.4723	0.84097
model_003			0.0001	0.5218	0.8984	0.5269	0.87705
model_004		16	0.01	0.4913	0.8707	0.4871	0.7426
model_005			0.001	0.4894	0.8648	0.4731	0.82113
model_006			0.0001	**0.5244**	**0.9081**	0.4982	0.86117
model_007	RMSProp	8	0.01	0.1495	0.3503	0.1094	0.35646
model_008			0.001	0.4528	0.8186	0.4451	0.81588
model_009			0.0001	0.4885	0.8688	0.4594	0.86547
model_010		16	0.01	0.0613	0.1089	0.066	0.10827
model_011			0.001	0.4429	0.8401	0.3832	0.83534
model_012			0.0001	0.4675	0.8367	0.4625	0.83549
model_013	SGD	8	0.01	0.5167	0.8916	0.5395	0.82795
model_014			0.001	0.5221	0.8673	**0.57**	0.8427
model_015			0.0001	0.5008	0.8559	0.5073	0.83361
model_016		16	0.01	0.508	0.8862	0.4757	0.86931
model_017			0.001	0.5217	0.8874	0.5654	**0.87751**
model_018			0.0001	0.5004	0.849	0.5268	0.84487

Table 5 shows how different hyperparameter combinations affect model performance, with model_006 achieving the best results with the Adam optimizer, batch size 16, and a learning rate of 0.0001, resulting in a mAP50-95 of 0.5244. This suggests that larger batch sizes and the Adam optimizer improve training stability and reduce overfitting, resulting in better PPE detection. Model_010 (RMSProp, batch size 16, learning rate 0.01) had the poorest performance (mAP50-95 of

0.0613), indicating that RMSProp is ineffective for YOLOv8. The next phase will concentrate on data augmentation to increase detection accuracy. Data augmentation will increase dataset diversity and generalization, resulting in more accurate detection of underrepresented PPE violations.

The Mosaic and MixUp augmentations were used to make the dataset more diverse and improve the model's ability to find PPE violations, especially in classes that aren't used very often, like "no-helmet" and "no-vest." To use mosaic augmentation, four training images were picked at random and put together to make a single 2×2 composite image. Each picture was resized to fit a quadrant on a 1280×1280 canvas, and the coordinates of the bounding boxes were changed to keep the accuracy of the annotations. This addition let the model see objects in a wider range of sizes, positions, and backgrounds, which helped it work better in a wider range of detection situations. The process of augmentation consistently increased the number of rare PPE violations, fixing imbalances in the dataset at both the spatial and contextual levels.

After Mosaic, MixUp augmentation was used to blend the composite with a fifth randomly selected image using a linear interpolation based on a Beta distribution ($\alpha = 0.4$) mixing coefficient. Merging the pixel values and bounding boxes from both images created synthetic examples that simulate transitional object boundaries and class co-occurrences. A Jupyter Notebook was used to implement this method in Python using OpenCV and NumPy. Custom scripts automatically read, transformed, and stored augmented samples and processed all images and annotations into a new augmented directory. The process added visually diverse and semantically balanced samples to the training set, improving model robustness and detection accuracy. The examples of images produced with this technique can be seen in Figure 4.

Figure 4 Mosaic and MixUp Augmentation Results

The model was then retrained on the augmented dataset and compared to previous models. The comparison focused on detection performance for each PPE class, evaluating improvements in mean Average Precision (mAP). The evaluation includes four model variations to determine the impact of augmentation and hyperparameter tuning. The first variation establishes a baseline by training a previous YOLO version (YOLOv5) with default hyperparameters for 100 epochs, as trained by a previous study on the same dataset [15]. The second model is the more recent YOLO version, YOLOv8, which was also trained with default settings to compare improvements between versions. The third model uses the best-performing hyperparameters identified during the hyperparameter tuning stage,

but without augmentation [13]. Finally, the fourth model employs the same optimized hyperparameters while incorporating Mosaic and MixUp augmentations to assess their impact on detection performance. The results are shown in Table 6.

Table 6 Model Performances Across Different Configurations

Model	*mAP50*	*mAP50-95*	*mAP50*				
			helmet	*no-helmet*	*no-vest*	*person*	*vest*
YOLOv5 Default Params [15]	0.867	0.5	0.936	0.675	0.838	**0.964**	0.922
YOLOv8 Default Params	0.9	0.515	**0.958**	0.856	0.845	0.952	0.888
YOLOv8 Best Params	0.888	0.52	0.936	0.715	**0.901**	0.961	0.929
YOLOv8 Best Params + Mosaic + MixUp [13]	**0.921**	**0.533**	0.943	**0.889**	0.866	0.957	**0,951**

Interestingly, although a previous study by Ciaglia et al. [15] reported that YOLOv5s trained on the same Construction Safety dataset (as part of Roboflow-100) achieved a mAP50 of 0.915, the reproduced baseline in this study only reached 0.867. Both models were trained under similar conditions, 640×640 resolution, 100 epochs, and default hyperparameters—but the performance discrepancy may be attributed to differences in training hardware, PyTorch versions, or slight preprocessing variations. Nonetheless, this study successfully closed the gap and outperformed the previously reported result by applying optimized training and augmentation techniques, with the final YOLOv8 model reaching a mAP50 of 0.921 and demonstrating superior performance across all PPE classes.

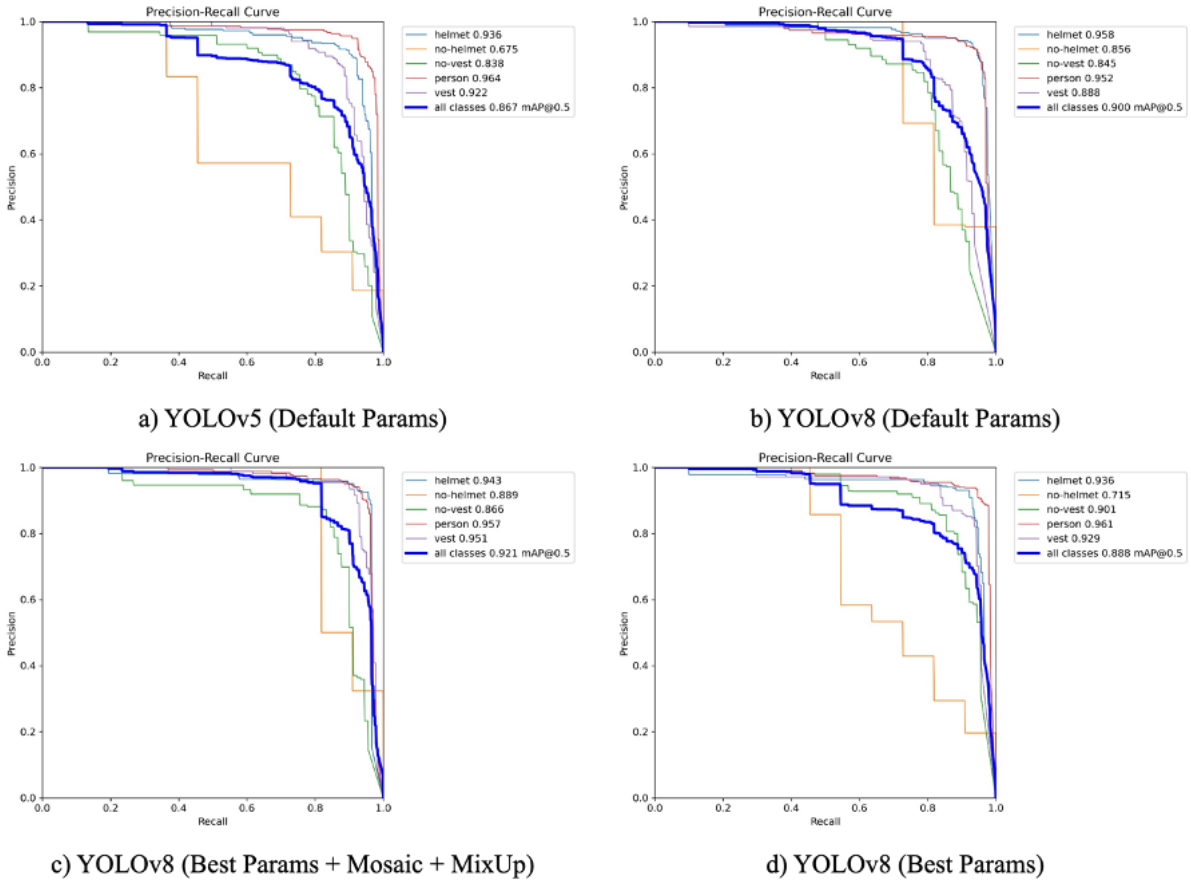

Figure 5 Precision-Recall Curve Across Each Model

Figure 5 presents the Precision-Recall (PR) curves, offering a comparative visualization of detection performance among various YOLO model configurations. The YOLOv5 model with default parameters (top-left) exhibits limited effectiveness in detecting underrepresented classes, such as no-helmet and no-vest, as evidenced by the shallow PR curves and sudden precision declines, which indicate low recall and high false-positive rates. In contrast, YOLOv8 with default parameters demonstrates a significant enhancement, particularly in helmet detection, as indicated by the pronounced and sustained PR curve. Nonetheless, the detection of individuals without helmets and vests remains moderate, indicating partial improvements in recall. The bottom-right configuration applies optimal hyperparameters without augmentation, enhancing overall detection, especially for no-vest, yet experiences reduced recall in the no-helmet class, indicating a precision-recall trade-off that leads to a more conservative model approach. The most significant performance is evident in the bottom-left plot, which illustrates YOLOv8 with optimized parameters and data augmentation. All classes, including the previously challenging no-helmet and no-vest categories, demonstrate high and stable precision across a broad recall range. This demonstrates enhanced generalization and model robustness, highlighting the essential function of augmentation in improving the detection of rare or visually subtle classes in practical safety applications.

The results clearly show how each step of the experiment, baseline comparison, hyperparameter tuning, and data addition, affects the model's overall performance. Going from YOLOv5 (mAP50: 0.867, mAP50-95: 0.500) to YOLOv8 with the default settings gave an immediate accuracy boost (mAP50: 0.900, mAP50-95: 0.515), especially for helmet (0.958) and no-helmet (0.856) detection.

These improvements show how much better YOLOv8's new architecture is at extracting finer spatial features than its predecessor.

The "YOLOv8 Best Params" model shows that hyperparameter tuning without adding anything else improved performance in the no-vest class (mAP50: 0.901). This shows that careful optimizer, learning rate, and batch size calibration is more important for some types of PPE. But this tuning caused mAP50 to drop slightly, from 0.900 to 0.888. This suggests that accuracy for some classes may have been lost because of changes in how learning works. This finding shows that hyperparameter optimization may not help all class categories equally in datasets that aren't balanced, even though it has a big effect.

After adding Mosaic and MixUp enhancements, the improvement happened. The final model was the most accurate overall (mAP50: 0.921, mAP50-95: 0.533), and it did a great job of finding violations like "no helmet" (mAP50: 0.889) and "vest" (mAP50: 0.951) that hadn't been picked up before. These results support the idea that adding more class diversity can make models more reliable and accurate without lowering their precision. The PR curves show that this is also true because they show high accuracy across a wide range of recalls for all PPE categories. In real life, this means that the model is not only more accurate, but it's also less likely to miss important violations. This makes it a good choice for enforcing PPE compliance in construction sites in real time.

4 Conclusion

Using YOLO-based models, this study looked at how hyperparameter tuning and data augmentation affected PPE detection performance. The best hyperparameter combination was identified using a grid search process, with the Adam optimizer, batch size of 16, and a learning rate of 0.0001, resulting in the highest mAP50-95 (0.5244) score. Mosaic and MixUp augmentations were used to improve detection performance by increasing dataset diversity and model generalization. The evaluation of various model variations revealed that augmentation was critical in improving recall and precision, especially for underrepresented PPE violations such as no-helmet and no-vest.

The best model was trained with the best hyperparameters and combined with Mosaic and MixUp augmentations. It had the highest mAP50 of 0.921 and the highest mAP50-95 of 0.533, doing better than both YOLOv5 and YOLOv8 without augmentation, showing that data augmentation improves detection performances. In terms of recall and precision, the Precision-Recall (PR) curves show that the model trained on augmented data does better than all classes. This is especially true for specific underrepresented classes, such as no-helmet and no-vest.

Future work should explore additional augmentation strategies, advanced loss reweighting techniques, and domain-specific fine-tuning to further optimize model robustness. Additionally, testing the model on real-world construction site footage and integrating it into a real-time PPE compliance monitoring system could provide valuable insights into its practical applications.

Bibliography

[1] Y. Adiratna *et al.*, *Profil Keselamatan dan Kesehatan Kerja Nasional Indonesia Tahun 2022*. Kementrian Ketenagakerjaan Republik Indonesia, 2022. Accessed: Sep. 05, 2024. [Online]. Available: https://satudata.kemnaker.go.id/publikasi/75

[2] W. S. Alaloul, S. Ammad, and S. Saad, “Health and Safety for Infrastructure Projects: PPE Adaptation and Barriers,” in *2020 2nd International Sustainability and Resilience Conference: Technology and Innovation*

in Building Designs, Institute of Electrical and Electronics Engineers Inc., Nov. 2020. doi: 10.1109/IEEECONF51154.2020.9319985.
[3] R. Ab Rahman, "Managing Safety at Work Issues in Construction Works in Malaysia: A Proposal for Legislative Reform," *Mod Appl Sci*, vol. 9, no. 13, p. 108, Nov. 2015, doi: 10.5539/mas.v9n13p108.
[4] A. K. Qisti, "Hubungan Tingkat Pengetahuan dan Sikap dengan Pemakaian Alat Pelindung Diri (APD) Pekerja di Pabrik PTPN7 Kabupaten Seluma," *Jurnal Sanitasi Profesional Indonesia*, vol. 2, no. 1, 2021, doi: 10.33088/jspi.v2i1.200.
[5] L. Ezzeddini *et al.*, "Analysis of the Performance of Faster R-CNN and YOLOv8 in Detecting Fishing Vessels and Fishes in Real Time," *PeerJ Comput Sci*, vol. 10, 2024, doi: 10.7717/PEERJ-CS.2033.
[6] A. Barlybayev *et al.*, "Personal Protective Equipment Detection Using YOLOv8 Architecture on Object Detection Benchmark Datasets: A Comparative Study," *Cogent Eng*, vol. 11, no. 1, 2024, doi: 10.1080/23311916.2024.2333209.
[7] M. Çiftçi, M. U. Türkdamar, and C. Öztürk, "Leveraging YOLO Models for Safety Equipment Detection on Construction Sites," *Journal of Computing Theories and Applications*, vol. 1, no. 4, pp. 492–506, May 2024, doi: 10.62411/jcta.10453.
[8] Ł. Popek, R. Perz, G. Galiński, and A. Abratański, "Optimization of Animal Detection in Thermal Images Using YOLO Architecture," *International Journal of Electronics and Telecommunications*, vol. 69, no. 4, pp. 825–831, 2023, doi: 10.24425/ijet.2023.147707.
[9] M. I. B. Ahmed *et al.*, "Personal Protective Equipment Detection: A Deep-Learning-Based Sustainable Approach," *Sustainability (Switzerland)*, vol. 15, no. 18, Sep. 2023, doi: 10.3390/su151813990.
[10] T. Mahendrakar *et al.*, "Performance Study of YOLOv5 and Faster R-CNN for Autonomous Navigation around Non-Cooperative Targets," in *2022 IEEE Aerospace Conference*, IEEE, 2022, p. 4265. doi: 10.1109/AERO53065.2022.9843537.
[11] P. Probst and B. Bischl, "Tunability: Importance of Hyperparameters of Machine Learning Algorithms," *Journal of Machine Learning Research*, vol. 20, pp. 1–32, 2019, doi: 10.48550/arXiv.1802.09596.
[12] R. Escobar Díaz Guerrero, L. Carvalho, T. Bocklitz, J. Popp, and J. L. Oliveira, "A Data Augmentation Methodology to Reduce the Class Imbalance in Histopathology Images," *Journal of Imaging Informatics in Medicine*, vol. 37, no. 4, pp. 1767–1782, Mar. 2024, doi: 10.1007/s10278-024-01018-9.
[13] N. Crasto, "Class Imbalance in Object Detection: An Experimental Diagnosis and Study of Mitigation Strategies," Mar. 2024, doi: 10.48550/arXiv.2403.07113.
[14] R. Padilla, S. L. Netto, E. A. B. Da Silva, and S. L. Netto, "A Survey On Performance Metrics For Object-Detection Algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020. doi: 10.1109/IWSSIP48289.2020.
[15] F. Ciaglia, F. S. Zuppichini, P. Guerrie, M. McQuade, and J. Solawetz, "Roboflow 100: A Rich, Multi-Domain Object Detection Benchmark," *arXiv preprint arXiv:2211.13523*, Nov. 2022, doi: 10.48550/arXiv.2211.13523.
[16] I. T. Young, J. J. Gerbrands, and L. J. Van, *Fundamentals of Image Processing*. Delft University of Technology, 1995.
[17] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On Empirical Comparisons of Optimizers for Deep Learning," *arXiv preprint arXiv:1910.05446*, Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.05446
[18] W. Zeng, "Image data augmentation techniques based on deep learning: A survey," 2024, *American Institute of Mathematical Sciences*. doi: 10.3934/mbe.2024272.
[19] R. Marco, S. S. S. Ahmad, and S. Ahmad, "An Improving Long Short Term Memory-Grid Search Based Deep Learning Neural Network for Software Effort Estimation," *International Journal of Intelligent Engineering and Systems*, vol. 16, no. 4, pp. 164–180, 2023, doi: 10.22266/ijies2023.0831.14.