# Final Project Report: PPE Detection Using YOLOv8

## 1. Introduction

In industrial and construction environments, ensuring worker safety is critical. According to the International Labour Organization (ILO), over 2.3 million people die annually due to work-related accidents or diseases, many of which are preventable by proper use of Personal Protective Equipment (PPE). Traditional manual inspection methods are time-consuming and prone to error. This project aims to develop an automated PPE detection system using state-of-the-art deep learning techniques, specifically leveraging the YOLOv8 object detection architecture

## 2. Problem Statement

The primary objective of the project is to automatically detect the presence or absence of PPE on individuals in images captured in industrial environments. This includes detecting 17 object classes related to safety equipment. The system should be robust, accurate, and capable of real-time deployment, with minimal latency and high generalizability. Key technical challenges include variations in lighting, object occlusion, background clutter, and severe class imbalance.

## 3. Dataset Description

* **Dataset Name:** SH17 Dataset for PPE Detection
* **Number of Images**: 8,099
* **Number of Classes**: 17
* **Data Format**: YOLO and VOC annotations
* **Source**: Pexels (under clear usage rights)

The dataset is diverse and high-resolution, with a significant imbalance among classes. Annotations were provided for all objects in YOLO format and were preprocessed for compatibility with the YOLOv8 framework.

## 4. Class Distribution Sample (Approximate):

* Helmet: 8,900 instances
* Gloves: 3,500 instances
* Face Shield: 450 instances
* Other PPE: < 300 instances

صورة تحتوي على نص, لقطة شاشة, رسم بياني, تخطيط

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.

**Class weights applied during training**

{0: 1.0, 1: 1.0, 2: 1.0, 3: 1.0, 4: 1.0, 5: 1.0, 6: 1.0, 7: 1.0, 8: 1.0, 9: 1.0, 10: 1.0, 11: 1.0, 12: 1.0, 13: 1.0, 14: 1.0, 15: 1.0, 16: 1.0}

Note: Equal weights were applied, and future work could involve class-based weighting derived from actual distribution

* Training Set: 6,479 images
* Validation Set: 1,620 images

## 5. Tools and Frameworks

* Deep Learning Framework: PyTorch
* Model: YOLOv8 via Ultralytics
* UI Framework: Streamlit
* Supporting Libraries: OpenCV, Albumentations, ONNX, Matplotlib, Seaborn

YOLOv8 was selected over YOLOv7 and Faster R-CNN due to its faster inference, better built-in support for augmentation, and streamlined integration with the Ultralytics training ecosystem. Below is a simplified comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Inference Speed** | **mAP Performance** | **Complexity** | **Suitability for Real-time** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | YOLOv7 |  |  |  |  | | Medium | |  | | --- | | High | | |  | | --- | | High | | Moderate |
| Faster R-CNN | Low | High | Very High | Low |
| YOLOv8 | High | Good | Moderate | High |

## 6. Data Preparation and Augmentation

The images were resized to 640x640 and normalized. Data augmentation was applied using Albumentations to include:

* Horizontal flips
* Rotation (±30 degrees)
* Random brightness/contrast
* CLAHE and HSV adjustments
* Random resized cropping

صورة تحتوي على شخص, وعاء, طبخ

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.

صورة تحتوي على شخص

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.

These augmentations aimed to improve generalization and address class imbalance. Future work may involve integrating Mosaic and CutMix strategies for further performance gains.

Bounding boxes were filtered to remove invalid annotations, ensuring compatibility with YOLO format.

## 7.Training Configuration and Results

* **Epochs**: 100
* **Total Time**: 18.87 hours
* **Hardware**: NVIDIA GTX 1660 Ti (6 GB VRAM), CUDA 12.8
* **Model**: YOLOv8s
* **Optimizer**: AdamW
* **Model Size**: 22.5MB (after optimizer strip)

**Inference Time**

* Preprocessing: 0.1ms
* Inference: 3.7ms
* Postprocessing: 1.0ms per image

## 8.Overall Performance

|  |  |
| --- | --- |
| Metric | Value |
| Precision | 0.673 |
| Recall | 0.448 |
| mAP@0.5 | 0.577 |
| mAP@0.5:0.95 | 0.384 |

## Class-wise Performance (Top and Low)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | mAP@0.5 | mAP@0.5:0.95 |
| Helmet | 0.872 | 0.839 | 0.889 | 0.756 |
| Gloves | 0.948 | 0.810 | 0.896 | 0.702 |
| Safety Belt | 0.931 | 0.830 | 0.903 | 0.729 |
| Face Shield | 0.852 | 0.732 | 0.831 | 0.575 |
| Other PPE | 0.500 | 0.361 | 0.418 | 0.253 |
| Goggles | 0.500 | 0.267 | 0.392 | 0.247 |
| Ear Protection | 0.316 | 0.040 | 0.182 | 0.113 |
| Safety Vest | 0.409 | 0.184 | 0.325 | 0.169 |

صورة تحتوي على نص, لقطة شاشة, رسم بياني, تخطيط

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.

Results show excellent detection for common PPE (helmet, gloves, safety belt), but significantly lower performance on rare or occluded items such as Ear Protection, Goggles, and Safety Vest. This highlights a persistent challenge with class imbalance and object visibility.

## 9. YOLOv8 Configuration

The dataset configuration was defined in a `data.yaml` file containing:

train: /kaggle/working/yolo\_dataset/train/images

val: /kaggle/working/yolo\_dataset/val/images

nc: 17

names: ['Helmet', 'Mask', 'Safety Vest', 'Gloves', 'Safety Glasses', 'Boots', 'Ear Protection', 'Harness', 'Coveralls', 'Respirator', 'Hard Hat', 'Face Shield', 'Safety Belt', 'Knee Pads', 'Reflective Tape', 'Goggles', 'Other PPE']

## 

## 10. Challenges Faced During Implementation

1. Class Imbalance: Several PPE classes were underrepresented, which led to biased predictions. This was mitigated using augmentation and class weighting.
2. Low Accuracy: The model struggled with certain classes due to limited examples and annotation inconsistencies.
3. Large Dataset Size: The dataset's high-resolution images increased training time and resource requirements.
4. High Computational Requirements: Training on dual T4 GPUs took approximately 7–8 hours. Real-time inference required further model optimization.
5. Poor Generalization on Real-World Images: Performance dropped in unseen scenarios due to limited domain diversity in training data.

## 11. Deployment

A lightweight web interface was developed using **Streamlit** to demonstrate real-time inference. The application allows image upload and visualizes detection results including:

* Image with bounding boxes
* A table showing detected class names and confidence scores

It supports both PyTorch and ONNX formats for deployment flexibility.

## 12. Environment Setup

Dependencies were defined in `requirements.txt`, which includes:

* `torch`, `torchvision`, `ultralytics`
* `onnx`, `onnxruntime`
* `opencv-python`, `albumentations`
* `streamlit`, `flask`

## 13. Future Improvements

* Model Optimization: Use of knowledge distillation or quantization for better edge deployment.
* Advanced Architectures: Experimenting with YOLOv8n + transformers for improved context understanding.
* Edge Deployment: Convert model to TensorRT or OpenVINO for faster local inference.
* Data Balancing: Apply oversampling, synthetic data generation for rare classes.
* Video Stream Integration: Extend Streamlit app to accept real-time video input.

## 13. Conclusion

This project demonstrates the application of YOLOv8 for PPE detection in industrial settings. Despite facing several technical challenges, the model achieved moderate performance and was successfully deployed in a web-based application for real-time usage. The system shows potential for integration in safety monitoring pipelines, with room for future optimization.

## 14. References

* Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv:1804.02767
* Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv:2004.10934
* Ultralytics YOLOv8 GitHub: https://github.com/ultralytics/ultralytics
* SH17 Dataset: https://www.kaggle.com/datasets/mughees/sh17-dataset-for-ppe-detection