Comparative Analysis of Architectures Based on You Only Look Once (YOLO) for the Detection of Personal Protective Equipments (PPE) on Construction Sites

A PROJECT REPORT

Submitted by

Tavish P [Reg No:RA2211003011190]

Pranav P [Reg No: RA2211003011175]

Dinessh V [Reg No: RA2211003011234]

Under the Guidance of

Dr. TYJ NAGA MALLESWARI

(Associate Professor, Department of Networking and Communications)

In partial fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

DEPARTMENT OF NETWORKING AND COMMUNICATIONS

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR -603203

APRIL 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 21CSE292P AIOT project report titled "**Comparative Analysis of Architectures Based on You Only Look Once (YOLO) for the Detection of Personal Protective Equipments (PPE) on Construction Sites**" is the bonafide work of **TAVISH P [RA2211003011190], PRANAV P [RA2211003011175] and DINESSH V [RA2211003011234]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

3dul2024

Dept. of Networking and Communications Kattankulathur 603 203 SRM IST

SIGNATURE

Dr. TY J NAGA MALLESWARI
Supervisor and Associate Professor
Dept. of Networking and Communications
SRM Institute of Science and Technology

SIGNATURE

Dr. K. ANNAPURANI
Professor and Head of the Department
Dept. of Network and Communications
SRM Institute of Science and Technology

SRM

Department of Networking and Communications

SRM Institute of Science & Technology

Own Work Declaration Form

Degree/ Course : **Bachelor of Technology in Computer Science and Engineering**

Student Name : **Tavish P, Pranav P, Dinessh V**

Registration Number : **RA2211003011190, RA2211003011175, RA2211003011234**

Title of Work : **Comparative Analysis of Architectures Based on You Only Look Once (YOLO) for the Detection of Personal Protective Equipments (PPE) on Construction Sites.**

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is my / our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g.fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

We are aware of and understand the University's policy on Academic misconduct and plagiarism and we certify that this assessment is our own work, except were indicated by referring, and that we have followed the good academic practices noted above.

Student 1 Signature: P.Tavish

Student 2 Signature:

Student 3 Signature:

Date: 30/4/2024

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our humble gratitude to Dr C. Muthamizhchelvan, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, Dr T.V.Gopal, for his invaluable support.

We wish to thank Dr Revathi Venkataraman, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, Dr. M. Pushpalatha, Professor and Head of the Department, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We register our immeasurable thanks to our Faculty Advisor, Dr. Nithyashri J and Dr. Kirubanantham P, Assistant Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, Dr. TYJ Naga Malleswari, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She provided me/us with the freedom and support to explore the research topics of our interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Networking and Communications Department, staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

TAVISH P [RA2211003011190]

PRANAV P [RA2211003011175]

DINESSH V [RA2211003011234

ABSTRACT

Ensuring the safety of workers on construction sites is paramount, and the detection of Personal Protective Equipment (PPE) plays a crucial role in preventing accidents and injuries. In recent years, deep learning-based object detection methods, particularly those using the You Only Look Once (YOLO) architecture, have shown remarkable performance in various applications, including PPE detection. This study presents a comparative analysis of two versions of the YOLO architecture, YOLOv8 and YOLOv9, for the detection of PPE on construction sites.

The YOLOv8 architecture, with its improvements over previous versions, offers enhanced performance in terms of speed and accuracy. By incorporating advanced features such as spatial pyramid pooling (SPP) and Path Aggregation Network (PAN), YOLOv8 achieves state-of-the-art results in object detection tasks. In this study, we train and evaluate a YOLOv8 model specifically tailored for PPE detection, analyzing its effectiveness in identifying safety helmets, vests, gloves, and other essential protective gear.

On the other hand, YOLOv9 represents the latest iteration of the YOLO architecture, introducing novel components and optimizations to further enhance detection performance. Leveraging techniques such as dynamic scaling and context aggregation, YOLOv9 aims to improve both accuracy and efficiency in object detection tasks. Through rigorous experimentation and evaluation, we assess the capabilities of YOLOv9 in detecting PPE items under varying conditions and complexities commonly encountered in construction environments.

The comparative analysis involves evaluating the detection accuracy, speed, and robustness of YOLOv8 and YOLOv9 models on a diverse dataset of construction site imagery. We investigate the impact of architectural differences and parameter settings on the performance of each model, providing insights into their strengths and limitations for PPE detection tasks. Additionally, we discuss practical considerations for deploying these models in real-world scenarios, including computational requirements and scalability.

The findings of this study contribute to the ongoing research efforts aimed at improving safety protocols in construction industries through the application of deep learning technologies. By identifying the strengths and weaknesses of YOLOv8 and YOLOv9 architectures for PPE detection, this research facilitates informed decision-making regarding the adoption of suitable models for enhancing workplace safety standards.

TABLE OF CONTENTS

List of Figures

LIST OF ABBREVATIONS

PPE	Personal Protective Equipment
YOLO	You Only Look Once
CNN	Convolutional Neural Networks
mAP	mean Average Precision
IDE	Integrated Development Environment
SMFD	Simulated Masked Face Dataset
ResNet	Residual Network
CSP	Cross Stage Connections
SPP	Spatial Pyramid Pooling
PAN	Pathy Aggregation Network
RNN	Recurrent Neural Network
NMS	Non-Maximum Suppression
SGD	Stochastic Gradient Descent
IOU	Intersection Over Union
FLOPs	Floating Point Operations Per second

CHAPTER 1

INTRODUCTION

The construction industry, characterized by its diverse tasks and hazardous working conditions, necessitates stringent safety measures to protect workers from potential harm. Personal Protective Equipment (PPE) stands as a crucial component of these safety measures, providing a shield against various occupational hazards such as falling objects, chemical exposure, and electrical hazards. However, ensuring the consistent and proper use of PPE among construction workers remains a persistent challenge. Manual monitoring methods, relying on periodic inspections and supervisor oversight, are often inadequate in ensuring comprehensive compliance, leading to potential gaps in safety protocols.

In response to these challenges, technological advancements in the field of computer vision offer promising avenues for automating PPE detection processes. By leveraging machine learning algorithms and real-time image analysis, automated PPE detection systems can offer a scalable and efficient solution to monitor PPE compliance on construction sites. These systems have the potential to revolutionize safety management practices by providing continuous monitoring, immediate feedback, and data-driven insights into compliance levels.

This comparative analysis focuses on evaluating the performance of two prominent object detection architectures, YOLOv8 and YOLOv9, in the context of PPE detection on construction sites. By examining the accuracy, speed, and robustness of these models, this study aims to provide valuable insights into their effectiveness in automating PPE detection tasks. Through a comprehensive evaluation, we seek to inform construction industry stakeholders about the feasibility and practical implications of adopting automated PPE detection systems as part of their safety management strategies. Ultimately, this research contributes to the ongoing efforts to enhance workplace safety standards in the construction industry through the integration of advanced technological solutions.

1.1 Background

In industrial sectors such as construction, where workers are exposed to a myriad of hazards, ensuring their safety is not just a legal requirement but a moral imperative. The construction industry consistently ranks among the most hazardous sectors, with risks ranging from falls and struck-by accidents to exposure to hazardous materials and machinery-related incidents. Personal Protective Equipment (PPE) serves as a critical line of defense against these hazards, providing workers with essential protective gear such as hard hats, safety goggles, gloves, and reflective vests.

Despite the clear benefits of PPE, enforcing its consistent use presents significant challenges. Manual monitoring of PPE compliance is labor-intensive and subject to human error, often resulting in lapses that compromise worker safety. Moreover, the dynamic nature of construction sites, characterized by changing environments and tasks, further complicates the task of ensuring PPE adherence. Traditional approaches to PPE enforcement, relying on periodic inspections and supervisor oversight, are inherently limited in their effectiveness and scalability.

The advent of advanced technologies, particularly in the realm of computer vision and artificial intelligence, offers a transformative solution to the challenge of PPE enforcement. By harnessing the power of machine learning algorithms and real-time image analysis, it becomes possible to automate the detection of PPE items worn by workers on construction sites. This automation not only streamlines the monitoring process but also enhances its accuracy and reliability, thus bolstering workplace safety outcomes. Automated PPE detection systems have the potential to revolutionize safety management practices in the construction industry, ushering in a new era of proactive risk mitigation and accident prevention.

1.2 Importance of Automated PPE Detection

The importance of automated PPE detection in the construction industry cannot be overstated, given its potential to revolutionize safety management practices and mitigate occupational hazards effectively. Firstly, automated PPE detection systems offer a proactive approach to safety management by providing real-time monitoring of PPE usage. This proactive approach enables immediate intervention in situations where PPE compliance is lacking, thereby reducing the risk of accidents and injuries before they occur. Additionally, by offering continuous monitoring capabilities, these systems can address the limitations of periodic inspections and supervisor oversight, ensuring comprehensive coverage of PPE compliance across construction sites.

Furthermore, automated PPE detection systems contribute to the optimization of resources and operational efficiency within construction projects. By automating the monitoring process, these systems alleviate the burden of manual inspections, freeing up valuable time and resources that can be redirected towards other critical tasks. Moreover, the integration of automated PPE detection into existing safety protocols enables construction site managers to make data-driven decisions regarding safety measures and resource allocation, leading to improved overall project efficiency and productivity.

Beyond the immediate benefits to construction site safety and operational efficiency, the implementation of automated PPE detection systems reflects a commitment to fostering a culture of safety and compliance within the construction industry. By prioritizing the well-being of workers and investing in innovative safety technologies, construction companies demonstrate their dedication to creating safer work environments and upholding their responsibilities towards employee welfare. This proactive approach not only enhances the reputation and credibility of construction firms but also fosters a positive workplace culture that prioritizes safety as a core value.

1.3 Introduction to You Only Look Once (YOLO) Architecture

The You Only Look Once (YOLO) architecture represents a significant advancement in the field of object detection, offering a unique approach that combines speed and accuracy for real-time detection tasks. Unlike traditional object detection methods that rely on complex multi-stage pipelines, YOLO adopts a single-stage detection approach, enabling it to process images rapidly without sacrificing accuracy. This is achieved through a unified convolutional neural network (CNN) architecture that predicts bounding boxes and class probabilities directly from the full image in a single pass.

One of the key innovations of YOLO is its grid-based approach to object detection, which divides the input image into a grid of cells and predicts bounding boxes and class probabilities for each cell. This grid-based approach allows YOLO to detect multiple objects within the same image efficiently, without the need for multiple passes or complex post-processing steps. Additionally, YOLO employs a loss function that directly optimizes detection performance, resulting in faster convergence and improved accuracy compared to traditional methods.

Since its inception, the YOLO architecture has undergone several iterations and refinements, with each version introducing improvements in terms of speed, accuracy, and robustness. YOLOv8 and YOLOv9, in particular, represent the latest iterations of the YOLO architecture, incorporating advancements in network architecture design, feature extraction, and optimization techniques. These improvements have further enhanced the performance of YOLO models,

making them well-suited for a wide range of real-world applications, including object detection in dynamic environments such as construction sites.

By providing a robust and efficient solution for object detection tasks, the YOLO architecture has gained widespread popularity and adoption across various industries, including surveillance, autonomous vehicles, and industrial automation. Its ability to deliver fast and accurate detections in real-time makes it particularly well-suited for applications that require rapid decision-making and response, such as PPE detection on construction sites. In the following sections, we delve into a comparative analysis of YOLOv8 and YOLOv9 architectures specifically in the context of PPE detection, aiming to provide insights into their respective strengths and weaknesses for this critical safety task.

1.4 Objectives of the Comparative Analysis

The primary objective of this comparative analysis is to provide a comprehensive evaluation of the performance of You Only Look Once (YOLO) architectures, specifically YOLOv8 and YOLOv9, in the context of detecting Personal Protective Equipment (PPE) on construction sites. By conducting a rigorous comparative analysis, we aim to achieve the following objectives:

1. Assess Performance Metrics: Evaluate the accuracy, speed, and robustness of YOLOv8 and YOLOv9 models in detecting various types of PPE items commonly worn by workers on construction sites. Performance metrics will include detection accuracy measured by mean Average Precision (mAP), inference speed measured in Frames Per Second (FPS), and other relevant metrics.
2. Identify Strengths and Weaknesses: Identify the strengths and weaknesses of each YOLO architecture in PPE detection tasks. This involves analyzing the ability of each model to accurately detect different PPE items under varying environmental conditions and levels of occlusion.
3. Compare Architectural Differences: Investigate architectural differences between YOLOv8 and YOLOv9 models and their impact on PPE detection performance. This includes examining architectural modifications, feature extraction techniques, and optimization strategies implemented in each version of the YOLO architecture.
4. Provide Practical Insights: Offer practical insights into the feasibility and efficacy of using YOLO-based models for automated PPE detection on construction sites. This involves discussing the practical implications of the comparative analysis results and providing recommendations for optimizing model performance and deployment in real-world scenarios.

Through these objectives, we aim to contribute valuable insights into the potential of YOLO architectures for enhancing workplace safety in the construction industry through automated PPE detection. By elucidating the strengths and limitations of each model, we seek to inform construction industry stakeholders and decision-makers about the suitability and practical considerations of adopting YOLO-based solutions for PPE compliance monitoring. Ultimately, this analysis aims to facilitate informed decision-making and contribute to the advancement of safety management practices in the construction industry.

1.5 Scope of the Study

The scope of this comparative analysis encompasses several key aspects related to the evaluation of You Only Look Once (YOLO) architectures for the detection of Personal Protective Equipment (PPE) on construction sites. Firstly, the study focuses on two specific versions of the YOLO architecture, namely YOLOv8 and YOLOv9, chosen based on their relevance and popularity in the field of computer vision.

In terms of datasets, the study utilizes annotated datasets specifically curated for PPE detection tasks in construction environments. These datasets encompass a diverse range of construction site scenarios, including varying lighting conditions, occlusions, and PPE configurations. Additionally, the study considers the availability and quality of training data as a crucial factor influencing the performance of the YOLO models.

The evaluation metrics employed in this study include traditional metrics such as mean Average Precision (mAP) for detection accuracy, as well as metrics related to inference speed and computational efficiency. By considering a comprehensive set of metrics, the study aims to provide a holistic assessment of the performance of YOLOv8 and YOLOv9 architectures in the context of PPE detection on construction sites.

Furthermore, the study takes into account practical considerations such as model deployment feasibility and scalability. This includes discussions on hardware requirements, model optimization techniques, and potential challenges associated with real-world deployment of automated PPE detection systems in construction settings.

While the primary focus of the study is on PPE detection in the construction industry, the insights and methodologies developed in this analysis may have broader implications for other industrial sectors requiring PPE compliance monitoring. However, it is essential to acknowledge the inherent limitations and constraints of the study, such as dataset bias, model generalizability, and environmental variability, which may impact the interpretation and applicability of the study findings.

CHAPTER 2

LITERATURE REVIEW

2.1 Deep learning neural network techniques

[1] In a study conducted by Delhi et al., the researchers applied a type of deep learning, which is computer vision, by recognizing the PPE on construction sites on an immediate basis. Accordingly, the researchers collected the dataset on which they conducted the research manually, in addition to applying web scraping. The dataset contained around 2500 images that were classified into four classes, as follows: NOHARDHAT, NOJACKET, SAFE, and NOT SAFE. Hence, YOLO-v3 was trained on that dataset. Furthermore, following the augmentation step based on the data, YOLO-v3 was trained on a sample of data. This gives the model resilience and generalization by performing flipping along with rotation on both sides, left and right, with an angle of 30 degrees. Further, by using a validation test strategy, the provided dataset was split into 90%, 8%, and 2% random segments for training, validation, and testing, respectively. Consequently, and based on the tested data, the model succeeded to fulfil an mAP and F1 score of 97%.

[2] Deep learning neural networks were applied in the research of Wang et al. for real-time detection and recognizing of objects to address the problem of worker safety by making sure that employees followed safety protocol. They consequently suggested applying YOLO-v3, YOLO-v4, and YOLO-v5, which are detectors based on deep learning of YOLO architectures. They used data from a high-quality dataset called CHV. Such data incorporated 1330 images extracted from Wang et al.'s dataset and broken down into six categories: person, vest, and helmets with four colors. The research results showed that YOLO-v5s had the fastest GPU performance of 52 FPS, while YOLO-v5x had the best mAP of 86.55%.

[3] A newly introduced cognitive analysis of safety measures for a monitoring system was proposed by Torrse et al. in another study. Such a system was used in this study to instantly determine whether personal protective equipment is being used appropriately based on data gathered by the monitoring of CCTV cameras. Further, the system employed a deep learning algorithm to identify objects. Hence, the study resulted in the creation of a YOLO-v4 system that could achieve an 80.19% mAP at 80 frames per second in real time. Most of the current deep learning detectors had limitations with far-away objects and close-range objects. YOLO models perform with a higher accuracy more than other detection models.

[4] A similar study by Hayat and Morgado Dias adopted a deep learning method for real time for the sake of identifying the heads and helmets on construction site workers. This paper investigated three different iterations of the well-known deep learning architecture YOLO: YOLO-v3, YOLO-v4, and YOLO-v5x. The model was implemented by the authors using the public dataset made available by Make ML. Therefore, a huge number of 3000 instances were used for training, and 1000 instances were used for testing. Furthermore, in this study, only the "Head" and "Helmet" were used as classes. To address the preprocessing of the images, power-law transformation was used for image preprocessing so as to increase the quality of contrast and lighting in such data. With accuracy, precision, recall, and F1 scores of 92%, 92.4%, 89.2%, and 90.8%, respectively, the YOLO-v5x model gave the best accuracy and, hence, the best performance.

[5] Gallo et al. suggested a system in to recognize personal protective equipment (PPE) in hazardous industrial areas. Deep neural networks were used for the system's analysis of a video stream. Five models—YOLO-v4, YOLO-v4-Tiny, SSD, CenterNet, EfficientDet, and MobileNet—were trained to determine whether or not the workers are implementing the safety measures by wearing safety equipment. The authors utilized three datasets: two were collected under controlled conditions and incorporated 215 and 236 images, respectively; the third one is an available public dataset with 7035 images. Because of its rapid detection speed, YOLO-v4-tiny was implemented in the system, achieving an mAP of 86%. Further, and by using the INRIA person dataset ,

[6] Li et al. were able to train an autonomous safety-helmet-wearing recognition system. Furthermore, a safety helmet detection model was suggested by Wang et al. , trained using 10,000 photos taken on construction sites by 10 distinct surveillance cameras. Geng et al. presented an enhanced helmet recognition method based on an unbalanced dataset of 7581 photos, the majority of which included a person wearing a helmet against a complicated backdrop. By testing it on 689 photos, it resulted in a label confidence of 0.982.

2.2 Transfer learning model-based automated technique

[7] A transfer learning model-based automated technique was developed by Vibhuti et al. to identify individuals who were not wearing masks in public in the period of the COVID-19 epidemic. InceptionV3, ResNet50, VGG16, MobileNet, MobileNetV2, and Xception were among the deep learning models that were employed in the intervention. Training, testing, and validation were conducted using the Simulated Masked Face dataset (SMFD). Through the use of fine-tuning strategy, the pretrained Inception (V3) model was developed and optimized. The greatest results, obtained with the SMFD dataset, were 100% accuracy and specificity in testing and 99.92% in training. The main outcomes highlighted the excellent accuracy of non-mask-wearer recognition automation achieved by the proposed transfer learning model.

2.3 Other Techniques

[8] Arunabha et al. proposed an enhanced YOLOv5 model based on DenseNet and the Swin-Transformer detection head, achieving commendable results in road damage detection.

[9] Jiang S, Zhou X et al. introduced the lightweight DWSC-YOLO model, incorporating DWS convolution and the Efficient attention mechanism, reducing the model size and making it apt for deployment on SAR radar devices.

[10] Sun C, Zhang S et al.presented the MCA-YOLOV5-Light model for safety helmet detection, embedding the MCA module and implementing sparse training.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 Hardware Requirements

The hardware specifications of the laptop used in our system architecture include:

- Processor: The laptop is equipped with a multi-core CPU (Central Processing Unit) to handle general-purpose computing tasks, including data preprocessing, model training, and inference. While not as powerful as dedicated server-grade CPUs, modern laptop processors offer sufficient performance for small to medium-sized machine learning workloads.
- Graphics Processing Unit (GPU): Some laptops may feature a dedicated GPU, typically from NVIDIA or AMD, to accelerate deep learning computations. GPUs are particularly beneficial for training deep learning models, such as YOLO architectures, as they can significantly reduce training times compared to CPU-only computations. However, the absence of a dedicated GPU does not preclude the use of laptops for developing and deploying PPE detection solutions, as model training can still be performed using CPU resources, albeit with longer training times.
- Memory (RAM): Sufficient RAM is essential for loading and processing large datasets, as well as for training and evaluating deep learning models. The laptop is equipped with an adequate amount of RAM to accommodate the data and computational requirements of the PPE detection tasks. Depending on the size of the datasets and complexity of the models, a minimum of 8 GB to 16 GB of RAM is typically recommended for smooth operation.
- Storage: The laptop includes built-in storage, such as SSDs (Solid State Drives) or HDDs (Hard Disk Drives), for storing datasets, code repositories, and trained model files. While SSDs offer faster read/write speeds and improved performance compared to HDDs, both storage types are suitable for storing the data and resources required for developing and deploying the PPE detection solution.
- Networking: The laptop is equipped with standard networking capabilities, including Wi-Fi and Ethernet connectivity, for accessing online resources, downloading datasets, and collaborating with team members. While not as powerful as enterprise-grade networking equipment, the built-in networking features of laptops suffice for most development and testing tasks.

3.2 Software Components

- Intel DevCloud & OneAPI: The Intel DevCloud provides access to a range of Intel hardware resources, including CPUs, GPUs, and FPGAs, enabling scalable and efficient computation for model training and inference. We leverage the OneAPI toolkit, which offers optimized libraries and frameworks for deep learning tasks, enhancing performance and accelerating development. By utilizing Intel DevCloud and OneAPI, we harness the computational power of Intel architectures to train and deploy our PPE detection models efficiently.
- JupyterLab: JupyterLab serves as our primary development environment, offering an interactive and collaborative workspace for model development, experimentation, and analysis. With its flexible interface and support for various programming languages, including Python, R, and Julia, JupyterLab enables seamless integration of code, visualizations, and documentation, facilitating an iterative and exploratory approach to model development.
- Amazon SageMaker Studio Lab: Amazon SageMaker Studio Lab provides a comprehensive set of tools and services for building, training, and deploying machine learning models in the cloud. We leverage SageMaker Studio Lab's integrated development environment (IDE) and managed infrastructure to streamline the end-to-end machine learning workflow, from data preparation and model training to deployment and monitoring. SageMaker Studio Lab offers built-in support for popular deep learning frameworks such as TensorFlow and PyTorch, simplifying the process of building and deploying YOLO-based PPE detection models.
- Google Colab: Google Colab provides a free, cloud-based platform for running Python code in a Jupyter notebook environment. We utilize Google Colab for collaborative model development and experimentation, leveraging its integration with Google Drive for seamless data access and sharing. With its support for GPU acceleration and pre-installed deep learning libraries, Google Colab enables us to train and evaluate YOLO models efficiently, even with large-scale datasets.
- Roboflow: Roboflow serves as a valuable resource for dataset management, annotation, and preprocessing. We utilize Roboflow's intuitive platform to upload, annotate, and augment construction site images, preparing them for training YOLO models. Roboflow offers a range of annotation tools, data augmentation techniques, and export options, making it easy to generate high-quality datasets tailored to our specific PPE detection task.

3.3 System Architecture

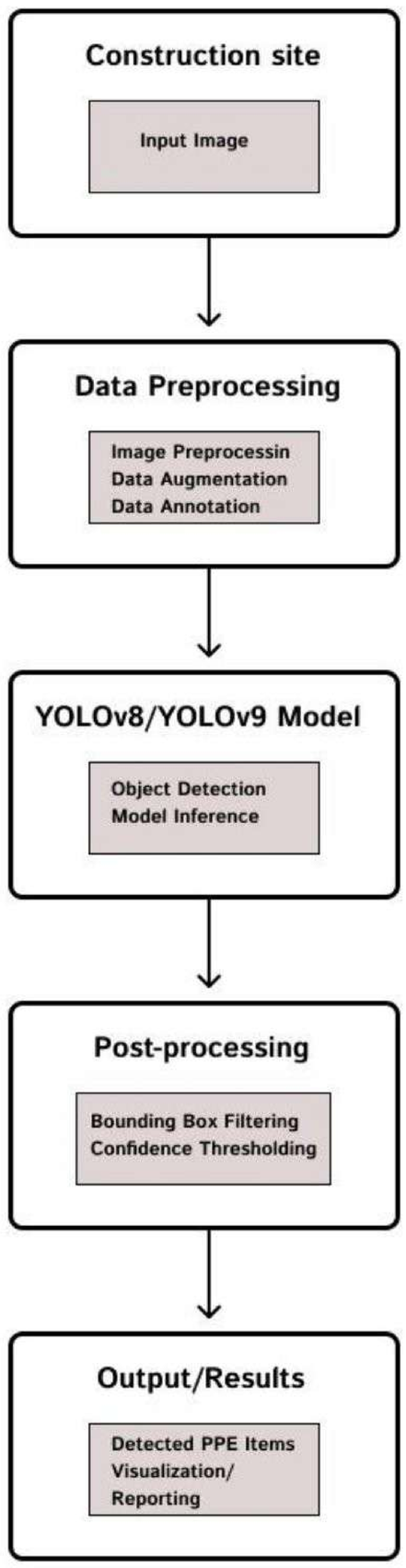


Fig3.1 System Architecture

The figure 3.1 depicts a system architecture for an automatic PPE detection system on a construction site using a YOLO model (e.g., YOLOv8 or YOLOv9). It outlines the steps involved in processing imagesfrom the construction site to generate reports on detected PPE items.

1. Construction Site: The process begins at the construction site, where surveillance cameras capture video footage as input for the system.
2. Data Preprocessing: The captured video footage enters a pre-processing stage to prepare it for the YOLO model.
    - Image Preprocessing: This step resizes and normalizes the images within the video frames to a format suitable for the model's input requirements.
    - Data Augmentation : This optional step can be used to artificially increase the diversity of the training data. It involves techniques like rotating, flipping, or modifying the brightness of existing images to generate new variations. This helps the model perform better on unseen real-world data.

3. YOLOv8/YOLOv9 Model: The preprocessed data is then fed into the core component, a pre-trained YOLO model (e.g., YOLOv8 or YOLOv9). This model is designed for object detection and can identify various PPE categories (e.g., hard hats, vests, safety glasses) within the images.
    - Object Detection: The YOLO model performs object detection on the images, recognizing and classifying PPE objects present in the scene.
    - Model Inference: This stage utilizes the trained model to make predictions on new incoming data (video frames).
4. Post-processing: After the YOLO model makes its detections, the results undergo further processing:
    - Bounding Box Filtering: This step might eliminate duplicate or low-quality bounding boxes proposed by the model.
    - Confidence Thresholding: Here, detections with a low confidence score (the model's certainty about the detection) are discarded. This ensures only high-confidence detections are considered for further processing.
5. Output/Results: The post-processed detections are then presented as the system's output:
    - Detected PPE Items: This represents the identified PPE categories (e.g., hard hats, vests) found in the video footage.
    - Visualization/Reporting: The system can generate visualizations by overlaying bounding boxes and class labels on the original video frames to show the detected PPE on the construction site. Additionally, it can produce reports summarizing the identified PPE items for further analysis.

Overall, the architecture represents a PPE detection system that leverages a YOLO model to automatically analyze images from a construction site, identify PPE usage, and generate reports to improve worker safety monitoring.

3.4 YOLO Architecture

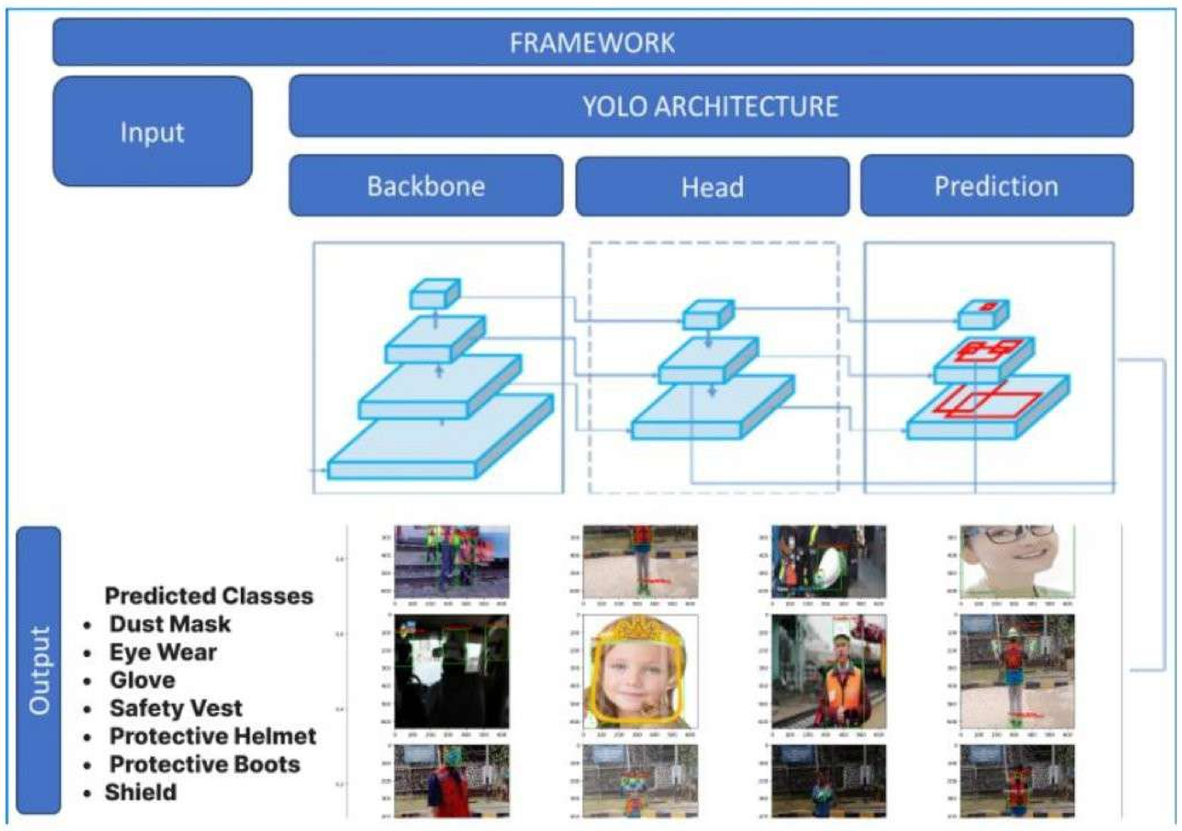


Fig 3.2 YOLOv8 Architecture

YOLO Architecture Principles:

- While specifics might differ, YOLOv8 likely follows core YOLO principles:
    - Single-Stage Detection: It predicts bounding boxes and class probabilities in a single network pass, making it fast for real-time applications.
    - Backbone-Head Structure: It likely separates the architecture into a Backbone for feature extraction (often a pre-trained CNN like Darknet or ResNet) and a Head for prediction using the extracted features.

Possible Components in a YOLOv8 PPE Detection System:

- Input: The system takes images or video frames from the construction site as input.
- Backbone: Extracts general features from the image (e.g., edges, shapes, colors).
- Neck : May be present for additional feature processing or fusion from different Backbone layers.
- Head: Uses features to predict bounding boxes and class probabilities for PPE categories (e.g., hard hat, vest).

Adaptation for PPE Detection:

- The YOLOv8 model would be pre-trained on a large dataset of images containing various PPE items used in construction.
- During training, the model learns to associate specific features in the images with corresponding PPE categories.

- When deployed, the model can then analyze new images and identify potential PPE objects based on the learned features.

Visualization:

- The system can overlay bounding boxes and class labels on the original images to visually represent the detected PPE items.

Overall, while the specific architecture details are unclear, YOLOv8 for PPE detection likely follows these general principles.

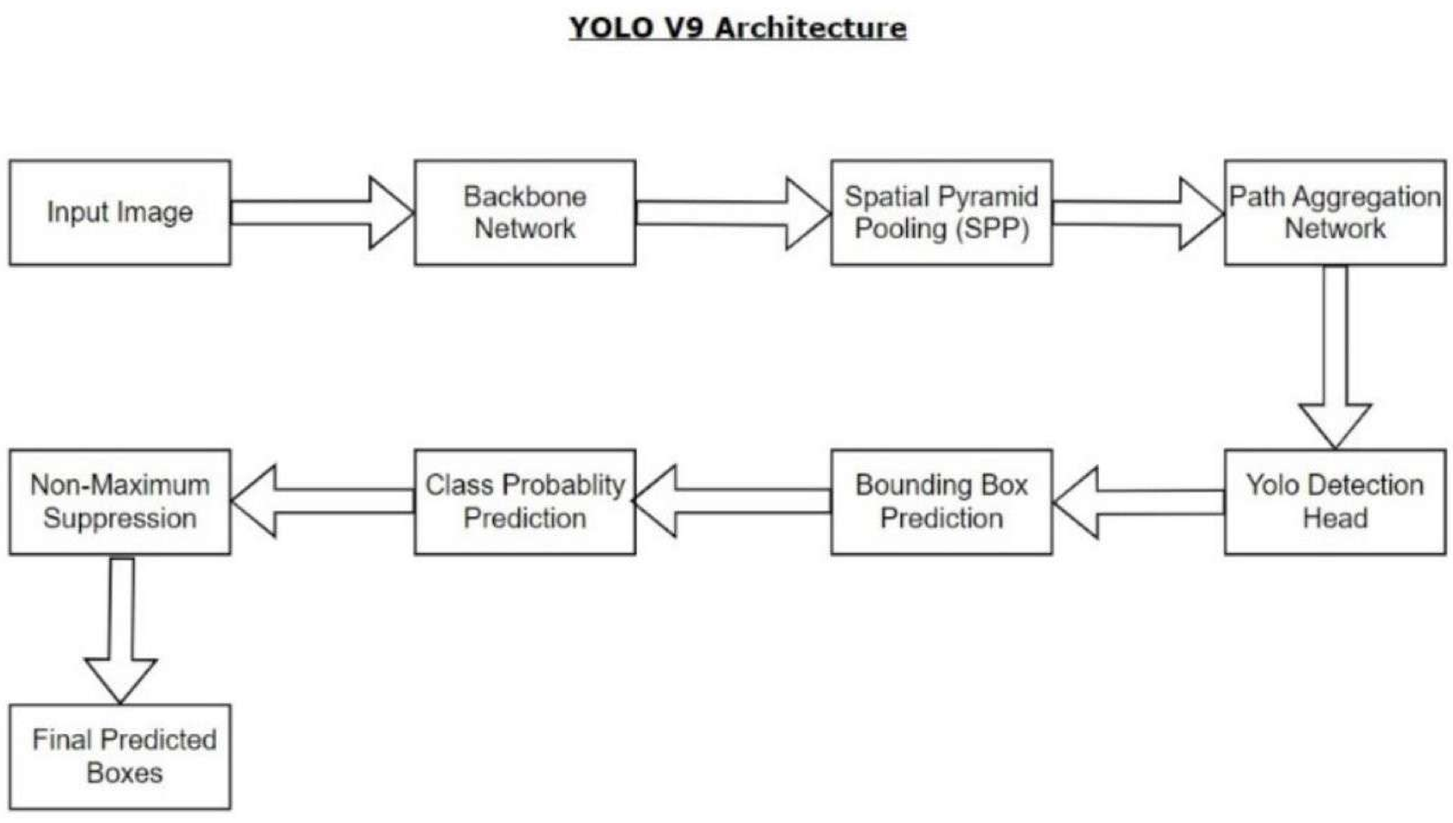


Fig 3.3 YOLOv9 Architecture

Key Components of the YOLOv9 Architecture for PPE Detection:

1. Input Image:
    - The input to the YOLOv9 model is an image containing the construction site scene with workers wearing PPE.
2. Backbone Network:
    - The Backbone Network, which could be CSPDarknet or EfficientNet, extracts hierarchical features from the input image.
3. Spatial Pyramid Pooling (SPP):
    - The SPP module captures contextual information at multiple scales without significantly increasing computational complexity, enhancing the model's ability to detect objects with varying spatial extents.
4. Path Aggregation Network (PAN):
    - The PAN modules aggregate features from different layers of the backbone network to improve object detection performance and feature representation.

5. YOLO Detection Head:
    - The YOLO Detection Head, integrated with PANet, refines feature representations and predicts bounding boxes and class probabilities for PPE items.
6. Bounding Box and Class Probability Prediction:
    - The model predicts bounding box coordinates and class probabilities for each detected PPE item.
7. Non-Maximum Suppression (NMS):
    - Non-Maximum Suppression is applied to remove redundant bounding boxes and retain only the most confident detections.
8. Final Predicted Boxes:
    - The final output consists of the predicted bounding boxes enclosing PPE items, ready for visualization or further analysis.

This architecture diagram (fig.3.3) illustrates the flow of information through the YOLOv9 model for PPE detection, highlighting the key stages of feature extraction, object detection, and post-processing.

CHAPTER 4

METHODOLOGY

In light of developing this study, we went through different phases to reach the optimum YOLO model in detecting PPE. In this study to compile an extensive collection for the literature review to guide and inform our investigation. Moreover, in order to guarantee relevant articles and lay the groundwork for an extensive literature review, the search criteria were carefully crafted. Consequently, after gaining knowledge from previous research and studying the limitations of other research, building up different YOLO models was our goal to reach. This was achieved using Intel Dev Cloud One API Jupyter Lab and Amazon Sagemaker Studio Lab.

The preparation of the dataset comprising both images and annotations was an important process to ensure that the results would be reliable. Subsequently, the stages of training and validating YOLO models were the highlighted phases. The model testing was performed by calculating performance evaluation metrics such as precision, recall, F1 score, and mAP.

In the final stages, a comparative analysis of different YOLO models was conducted by analyzing the results to draw meaningful conclusions and contribute to the evolving field of detecting PPEs.

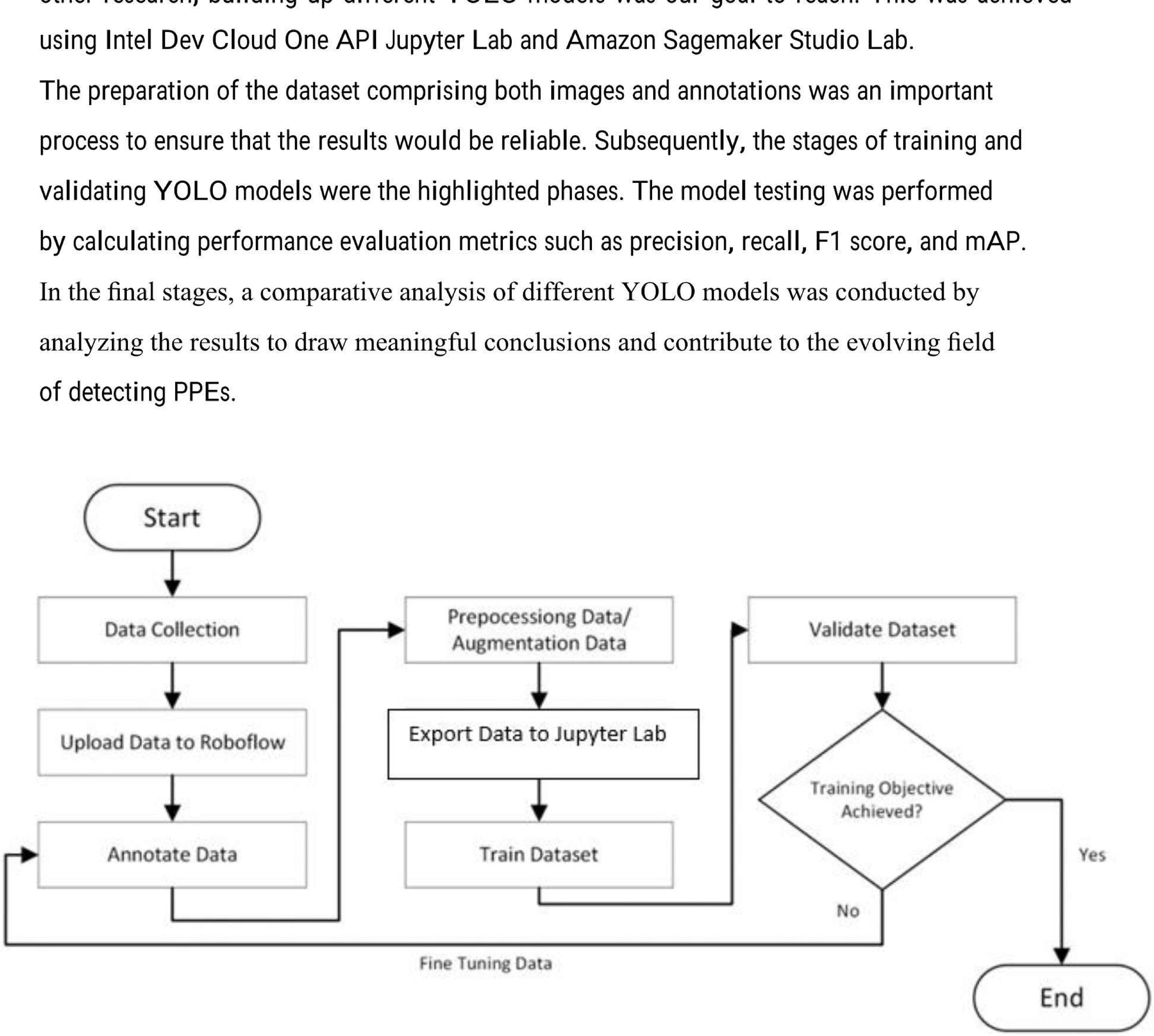


Fig. 4.1 Data flow Diagram

4.1 Dataset Acquisition

In the process of dataset acquisition, we utilize the robust resources provided by Roboflow, a platform designed for efficient management and annotation of image datasets. Roboflow offers a comprehensive repository of annotated image datasets, including those specifically curated for PPE detection tasks in construction environments. Leveraging Roboflow's extensive library ensures access to diverse and high-quality datasets, covering a wide range of construction site scenarios and PPE usage scenarios.

Through the Roboflow platform, we import annotated image datasets tailored to our specific requirements for PPE detection on construction sites. These datasets are carefully curated to include images captured from various construction sites with different environmental conditions, lighting variations, and PPE configurations. By utilizing datasets sourced from Roboflow, we benefit from the platform's rigorous quality control measures and annotation standards, ensuring the accuracy and consistency of the annotated data.

Furthermore, Roboflow offers powerful annotation tools and preprocessing capabilities, facilitating efficient data preparation processes. Annotations such as bounding boxes indicating the locations of PPE items are seamlessly integrated into the imported datasets, streamlining the model training pipeline. Additionally, Roboflow provides data augmentation functionalities, allowing for the generation of augmented images to enhance dataset diversity and model robustness.

By leveraging Roboflow's resources and capabilities for dataset acquisition, we expedite the data preparation process and ensure access to high-quality annotated datasets tailored to our specific research objectives. This enables us to focus our efforts on model training, evaluation, and comparative analysis, ultimately advancing the development of automated PPE detection systems for construction site safety management.

```
from roboflow import Roboflow
rf = Roboflow(api_key="F37pg4gr42c1PivmIqV0")
project = rf.workspace("project-uyrxf").project("ppe_detection-v1x3l")
version = project.version(2)
dataset = version.download("yolov8")
```

```
loading Roboflow workspace...
loading Roboflow project...
Dependency ultralytics==8.0.196 is required but found version=8.1.27, to fix: `pip install ultralytics==8.0.196`
Downloading Dataset Version Zip in PPE_Detection-2 to yolov8:: 100%|██████████| 198176/198176 [00:54<00:00, 3605.20it/s]

Extracting Dataset Version Zip to PPE_Detection-2 in yolov8:: 100%|██████████| 6472/6472 [00:18<00:00, 356.30it/s]
```

Fig. 4.2 Importing Dataset from Roboflow

4.2 Data Preprocessing

Data preprocessing is a critical step in preparing the dataset for training the YOLOv8 and YOLOv9 models. This section outlines the image processing, data augmentation, and data annotation techniques used to enhance the quality and diversity of the dataset.

4.2.1 Image Processing:

Image processing techniques are applied to standardize and enhance the quality of the input images before feeding them into the model. Common image processing steps include:

1. Resizing: Resize images to a uniform size compatible with the input dimensions of the YOLOv8 and YOLOv9 models. Maintaining consistent image dimensions ensures efficient processing during training.
2. Normalization: Normalize pixel values to a common scale (e.g., [0, 1]) to improve model convergence and stability during training. Standard normalization techniques include min-max scaling or z-score normalization.
3. Color Space Conversion: Convert images to a specific color space (e.g., RGB, grayscale) depending on the requirements of the model and application. Consistent color representation simplifies model training and improves performance.

4.2.2 Data Augmentation:

Data augmentation techniques are applied to artificially increase the diversity and variability of the dataset, thereby improving the model's robustness and generalization capabilities. Common data augmentation techniques include:

1. Random Cropping: Randomly crop regions of the image to simulate variations in object placement and scene composition. Cropping introduces spatial variations and helps the model learn to detect objects at different locations within the image.
2. Rotation: Rotate images by a random angle to simulate variations in object orientation. Rotation augmentation helps the model learn to detect objects from different viewpoints and angles.
3. Horizontal Flipping: Flip images horizontally with a certain probability to simulate mirror reflections. Horizontal flipping introduces lateral variations and helps the model learn invariant features.
4. Brightness and Contrast Adjustment: Adjust the brightness and contrast of images to simulate variations in lighting conditions. Brightness and contrast augmentation helps the model learn to detect objects under different illumination levels.

4.2.3 Data Annotation

Data were annotated using Roboflow which involves labeling objects of interest in images with bounding boxes and corresponding class labels using the Roboflow platform. Here's an overview of the data annotation process in Roboflow:

1. Upload Images:
    - Begin by uploading the images containing objects that need to be annotated to the Roboflow platform. You can upload images individually or in batches, depending on your dataset size.
2. Create Dataset:
    - After uploading the images, create a new dataset in Roboflow to organize and manage the annotated data. Give the dataset a descriptive name to easily identify its contents.
3. Annotate Objects:
    - Select the dataset you created and navigate to the annotation tool within Roboflow. This tool allows you to annotate objects by drawing bounding boxes around them in the images.
    - Click on the object of interest in the image and drag to create a bounding box around it. Adjust the size and position of the bounding box as needed to accurately encapsulate the object.
4. Assign Class Labels:
    - Once the bounding box is drawn, assign a class label to the annotated object to indicate its category (e.g., helmet, vest). Roboflow provides options to define custom class labels based on your specific application requirements.
5. Save Annotations:
    - After annotating all objects in the images, save the annotations within the Roboflow platform. The annotations are stored in a standardized format compatible with various machine learning frameworks and annotation formats.
6. Export Annotated Data:
    - Once annotations are saved, export the annotated data from Roboflow in the desired format (e.g., YOLO, COCO, Pascal VOC). Roboflow supports a wide range of annotation formats, making it easy to integrate annotated data into different machine learning pipelines.
7. Review and Iterate:
    - Review the annotated data to ensure accuracy and completeness. Iterate on the annotation process as needed to refine and improve the quality of annotations.

4.3 Training and Validation

Training and validation are integral components of the model development process, ensuring that the trained models achieve high accuracy and robustness in detecting PPE items on construction sites. This section outlines the training and validation procedures employed to optimize model performance and evaluate generalization capabilities.

4.3.1 Dataset Splitting:

- The annotated dataset is divided into three subsets: training, validation, and test sets, using a predefined ratio (70-20-10) (Figure 4.3 and 4.4).
- The training set is used to train the model parameters, while the validation set is utilized to tune hyperparameters and monitor model performance during training.
- The test set, held out from model training and hyperparameter tuning, serves as an independent evaluation dataset to assess the generalization of the trained models.

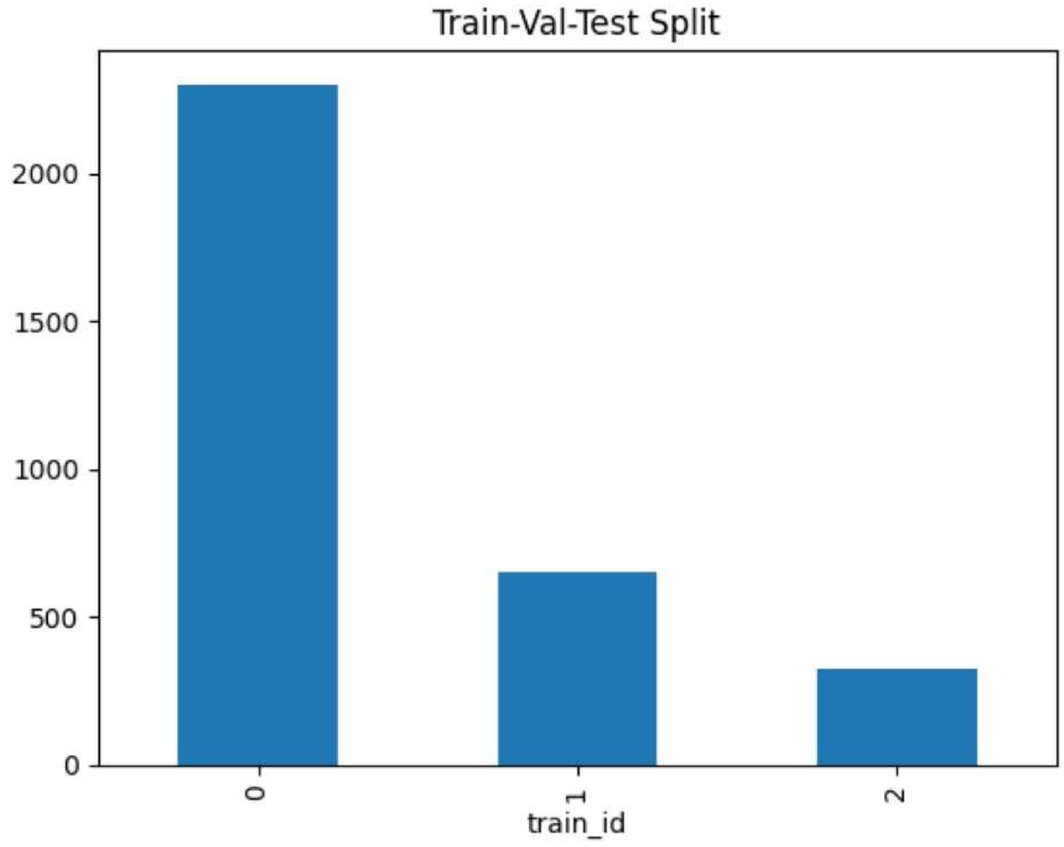


Fig. 4.3 Train-Valid-Test Split

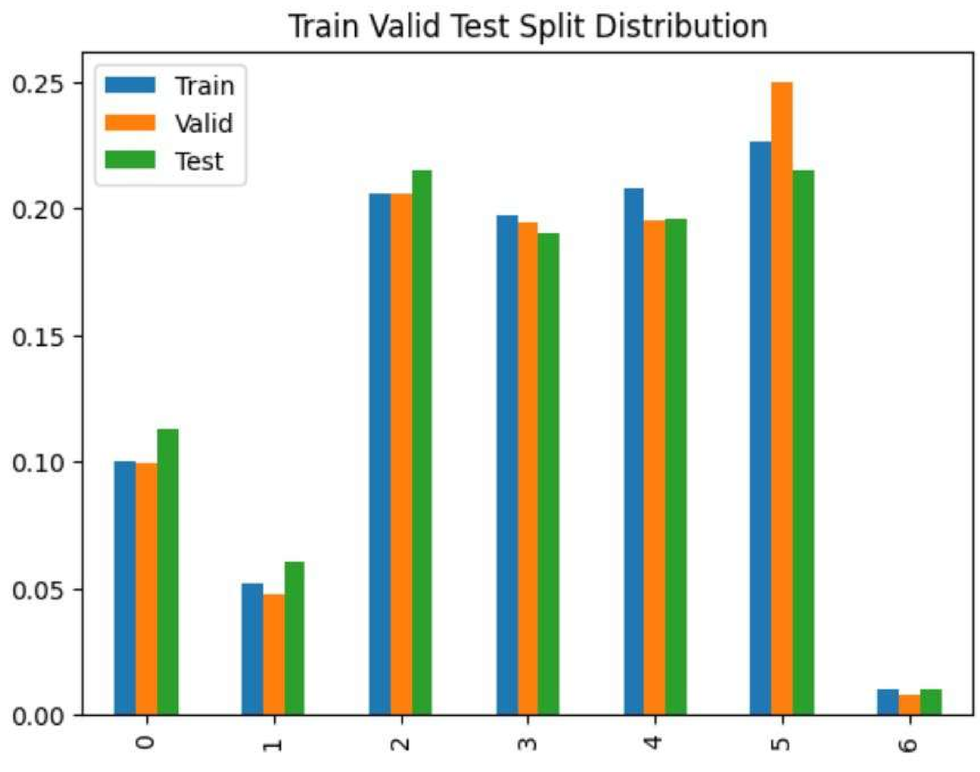


Fig. 4.4 Train-Valid-Test Split Distribution based on classes

4.3.2 Training Procedure:

- Batch Training: Training is conducted in batches, where a subset of the training data is processed in each iteration. Batch training enables efficient utilization of computational resources and facilitates gradient-based optimization.
- Loss Calculation: The loss function, typically a combination of localization and classification losses (e.g., YOLO loss), is computed during each training iteration to measure the discrepancy between predicted and ground truth bounding boxes.
- Backpropagation: Gradients of the loss function with respect to the model parameters are computed using backpropagation, and the model parameters are updated accordingly using an optimization algorithm such as stochastic gradient descent (SGD) or Adam.
- Epochs: Training proceeds over multiple epochs, where each epoch corresponds to a complete pass through the training dataset. Multiple epochs allow the model to iteratively refine its parameters and improve detection accuracy.

```
[*]: !yolo task=detect mode=train epochs=50 data='/home/u215978/new yolov8/PPE_Detection-2/data.yaml' model=yolov8n.pt imgsz=640

New https://pypi.org/project/ultralytics/8.2.2 available 😃 Update with 'pip install -U ultralytics'
Ultralytics YOLOv8.1.27 🚀 Python-3.9.18 torch-2.2.1+cu121 CPU (Intel Xeon Gold 6128 3.40GHz)
engine/trainer: task=detect, mode=train, model=yolov8n.pt, data=/home/u215978/new yolov8/PPE_Detection-2/data.yaml, epochs=50, time=None, patience=100,
batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=8, project=None, name=train2, exist_ok=False, pretrained=True, optimi
zer=auto, verbose=True, seed=0, deterministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=1.0,
profile=False, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True, split=val, save_json=False, save_hybrid=False, c
onf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None, vid_stride=1, stream_buffer=False, visualize=False, augment=False, agno
stic_nms=False, classes=None, retina_masks=False, embed=None, show=False, save_frames=False, save_txt=False, save_conf=False, save_crop=False, show_lab
els=True, show_conf=True, show_boxes=True, line_width=None, format=torchscript, keras=False, optimize=False, int8=False, dynamic=False, simplify=False,
opset=None, workspace=4, nms=False, lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.
1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smoothing=0.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.
5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0, auto_augment=randaugment, erasing=0.4, crop_fraction=1.0,
cfg=None, tracker=botsort.yaml, save_dir=runs/detect/train2
Overriding model.yaml nc=80 with nc=7

                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 22        [15, 18, 21]  1    752677  ultralytics.nn.modules.head.Detect           [7, [64, 128, 256]]
Model summary: 225 layers, 3012213 parameters, 3012197 gradients, 8.2 GFLOPs

Transferred 319/355 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/detect/train2', view at http://localhost:6006/
Freezing layer 'model.22.dfl.conv.weight'
train: Scanning /home/u215978/new yolov8/PPE_Detection-2/train/labels.cache... 2
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=0.01, clip_limit=(1, 4.0), tile_grid_si
ze=(8, 8))
val: Scanning /home/u215978/new yolov8/PPE_Detection-2/valid/labels.cache... 637
Plotting labels to runs/detect/train2/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automatically...
optimizer: AdamW(lr=0.000909, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias(decay=0.0)
TensorBoard: model graph visualization added ✅
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs/detect/train2
Starting training for 50 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/50         0G      1.617      4.235      1.568        110        640:
```

Fig. 4.5 Yolov8 Training Procedure

```
[ ]: !python /home/studio-lab-user/Yolov9/yolov9/train_dual.py --workers 8 --device 0 --batch 8 --data '/home/studio-lab-user/Yolov9/PPE_Detection-2/data.
```

```
train_dual: weights=/home/studio-lab-user/Yolov9/yolov9-e.pt, cfg=/home/studio-lab-user/Yolov9/yolov9/models/detect/yolov9-e.yaml, data=/home/studio
-lab-user/Yolov9/PPE_Detection-2/data.yaml, hyp=/home/studio-lab-user/Yolov9/yolov9/data/hyps/hyp.scratch-high.yaml, epochs=50, batch_size=8, imgsz=
640, rect=False, resume=False, nosave=False, noval=False, noautoanchor=False, noplots=False, evolve=None, bucket=, cache=None, image_weights=False,
device=0, multi_scale=False, single_cls=False, optimizer=SGD, sync_bn=False, workers=8, project=runs/train, name=yolov9-e-finetuning, exist_ok=Fals
e, quad=False, cos_lr=False, flat_cos_lr=False, fixed_lr=False, label_smoothing=0.0, patience=100, freeze=[0], save_period=-1, seed=0, local_rank=-
1, min_items=0, close_mosaic=15, entity=None, upload_dataset=False, bbox_interval=-1, artifact_alias=latest
YOLO 🚀 v0.1-86-g1bbce4d Python-3.9.16 torch-2.2.2+cu121 CUDA:0 (Tesla T4, 14931MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=
0.5, cls_pw=1.0, obj=0.7, obj_pw=1.0, dfl=1.5, iou_t=0.2, anchor_t=5.0, fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1,
scale=0.9, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.15, copy_paste=0.3
ClearML: run 'pip install clearml' to automatically track, visualize and remotely train YOLO 🚀 in ClearML
Comet: run 'pip install comet_ml' to automatically track and visualize YOLO 🚀 runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/

                 from  n    params  module                                  arguments
  0                -1  1         0  models.common.Silence                   []
  1                -1  1      1856  models.common.Conv                      [3, 64, 3, 2]
  2                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  3                -1  1    252160  models.common.RepNCSPELAN4              [128, 256, 128, 64, 2]
  4                -1  1    164352  models.common.ADown                     [256, 256]
  5                -1  1   1004032  models.common.RepNCSPELAN4              [256, 512, 256, 128, 2]
  6                -1  1    656384  models.common.ADown                     [512, 512]
  7                -1  1   4006912  models.common.RepNCSPELAN4              [512, 1024, 512, 256, 2]
  8                -1  1   2623488  models.common.ADown                     [1024, 1024]
  9                -1  1   4269056  models.common.RepNCSPELAN4              [1024, 1024, 512, 256, 2]
 10                 1  1      4160  models.common.CBLinear                  [64, [64]]
 11                 3  1     49344  models.common.CBLinear                  [256, [64, 128]]
 12                 5  1    229824  models.common.CBLinear                  [512, [64, 128, 256]]
 13                 7  1    984000  models.common.CBLinear                  [1024, [64, 128, 256, 512]]
 31          [-1, 7]  1         0  models.common.Concat                    [1]
 32                -1  1   4005888  models.common.RepNCSPELAN4              [1536, 512, 512, 256, 2]
 33                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 34          [-1, 5]  1         0  models.common.Concat                    [1]
 35                -1  1   1069056  models.common.RepNCSPELAN4              [1024, 256, 256, 128, 2]
 36                28  1    787968  models.common.SPPELAN                   [1024, 512, 256]
 37                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 38         [-1, 25]  1         0  models.common.Concat                    [1]
 39                -1  1   4005888  models.common.RepNCSPELAN4              [1536, 512, 512, 256, 2]
 40                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 41         [-1, 22]  1         0  models.common.Concat                    [1]
 42                -1  1   1069056  models.common.RepNCSPELAN4              [1024, 256, 256, 128, 2]
 43                -1  1    164352  models.common.ADown                     [256, 256]
 44         [-1, 39]  1         0  models.common.Concat                    [1]
 45                -1  1   3612672  models.common.RepNCSPELAN4              [768, 512, 512, 256, 2]
 46                -1  1    656384  models.common.ADown                     [512, 512]
 47          [-1, 36]  1         0  models.common.Concat                    [1]
 48                -1  1  12860416  models.common.RepNCSPELAN4              [1024, 512, 1024, 512, 2]
 49[35, 32, 29, 42, 45, 48]  1  10992074  models.yolo.DualDDetect                 [7, [256, 512, 512, 256, 512, 512]]
yolov9-e summary: 1475 layers, 69417098 parameters, 69417066 gradients, 244.9 GFLOPs

Transferred 2160/2172 items from /home/studio-lab-user/Yolov9/yolov9-e.pt
AMP: checks passed ✅
optimizer: SGD(lr=0.01) with parameter groups 356 weight(decay=0.0), 375 weight(decay=0.0005), 373 bias
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=0.01, clip_limit=(1, 4.0), tile_grid
_size=(8, 8))
train: Scanning /home/studio-lab-user/Yolov9/PPE_Detection-2/train/labels.cache.
val: Scanning /home/studio-lab-user/Yolov9/PPE_Detection-2/valid/labels.cache...
Plotting labels to runs/train/yolov9-e-finetuning3/labels.jpg...
Image sizes 640 train, 640 val
Using 4 dataloader workers
Logging results to runs/train/yolov9-e-finetuning3
Starting training for 50 epochs...
```

Fig. 4.6 YOLOv9 Training Procedure

Figures 4.5 and 4.6 illustrate the training procedures for YOLOv8 and YOLOv9, showcasing the steps and considerations involved in training these models for object detection tasks.

4.3.3 Validation Procedure:

- Model Evaluation: During training, the model's performance is periodically evaluated on the validation set to monitor its progress and prevent overfitting.
- Metrics Calculation: Evaluation metrics such as mean Average Precision (mAP), precision, recall, and F1 score are calculated on the validation set to assess detection accuracy and robustness.
- Early Stopping: Training may be halted early if validation performance fails to improve over a predefined number of epochs, indicating potential overfitting. Early stopping prevents the model from memorizing the training data and encourages generalization to unseen examples.

```
[5]: !yolo task=detect mode=val model='/home/u215978/new yolov8/runs/detect/train/weights/best.pt' data = '/home/u215978/new yolov8/PPE_Detection-2/data.yaml

Ultralytics YOLOv8.1.27 🚀 Python-3.9.18 torch-2.2.1+cu121 CPU (Intel Xeon Gold 6128 3.40GHz)
Model summary (fused): 168 layers, 3007013 parameters, 0 gradients, 8.1 GFLOPs
val: Scanning /home/u215978/new yolov8/PPE_Detection-2/valid/labels.cache... 637
                 Class     Images  Instances      Box(P          R      mAP50  m
                   all        637       3064      0.905      0.835      0.893      0.625
             Dust Mask        637        308      0.974      0.916      0.971      0.678
              Eye Wear        637        147       0.88        0.7      0.821      0.473
                 Glove        637        631      0.976      0.889      0.937       0.67
      Protective Boots        637        608       0.97      0.951      0.984      0.699
     Protective Helmet        637        753      0.958       0.95      0.985      0.753
           Safety Vest        637        592      0.877      0.878      0.929      0.715
                Shield        637         25      0.698       0.56      0.628      0.385
Speed: 1.5ms preprocess, 38.6ms inference, 0.0ms loss, 0.6ms postprocess per image
Results saved to runs/detect/val4
💡 Learn more at https://docs.ultralytics.com/modes/val
```

Fig. 4.7 YoloV8 Validation Procedure

```
!python /content/drive/MyDrive/newyolov9/yolov9/val_dual.py --data '/content/drive/MyDrive/newyolov9/PPE_Detection-2/data.yaml' --img 640 --batch 16 --conf 0.001 --iou 0.7 --device 0

val_dual: data=/content/drive/MyDrive/newyolov9/PPE_Detection-2/data.yaml, weights=['/content/drive/MyDrive/newyolov9/best (1).pt'], batch_size=16, imgsz=640, conf_thres=0.001, iou_thr
YOLO 🚀 v0.1-89-g93f1a28 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
yolov9-e summary: 839 layers, 68557066 parameters, 0 gradients, 240.7 GFLOPs
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
100% 755k/755k [00:00<00:00, 24.7MB/s]
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork() was called. os.fork() is incompatible with multithreaded code, and JAX is multithreaded, so this will li
  self.pid = os.fork()
val: Scanning /content/drive/MyDrive/newyolov9/PPE_Detection-2/valid/labels... 637 images, 0 backgrounds, 0 corrupt: 100% 637/637 [00:09<00:00, 68.83it/s]
val: New cache created: /content/drive/MyDrive/newyolov9/PPE_Detection-2/valid/labels.cache
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 40/40 [01:00<00:00,  1.51s/it]
                   all        637       3064      0.926      0.884       0.93      0.667
             Dust Mask        637        308      0.969      0.945      0.969      0.706
              Eye Wear        637        147      0.868       0.81      0.854      0.491
                 Glove        637        631      0.963      0.897      0.955      0.711
      Protective Boots        637        608      0.977      0.961      0.991      0.729
     Protective Helmet        637        753      0.966      0.953      0.984       0.78
           Safety Vest        637        592      0.878      0.889      0.936      0.726
                Shield        637         25      0.859      0.731      0.821      0.524
Speed: 0.2ms pre-process, 78.1ms inference, 4.4ms NMS per image at shape (16, 3, 640, 640)

Evaluating pycocotools mAP... saving runs/val/yolov9_ppe_c_640_val/best (1)_predictions.json...
loading annotations into memory...
pycocotools unable to run: [Errno 2] No such file or directory: '/content/drive/MyDrive/newyolov9/yolov9/annotations/instances_val2017.json'
Results saved to runs/val/yolov9_ppe_c_640_val
```

Fig. 4.8 YoloV9 Validation Procedure

Figures 4.7 and 4.8 illustrate the validation procedure for the best.pt model of YOLOv8 and YOLOv9. The validation process is crucial for assessing the performance of these models on unseen data, ensuring their accuracy and reliability in real-world applications.

4.4 Metrices for Performance Evaluation

The dataset was benchmarked by using state-of-the-art one-stage object detection models. The CHV dataset was tested by different YOLO models such as YOLO-v5x, YOLO-v5l, YOLO-v5m, YOLO-v5s, YOLO-v5n, YOLO-v5x6, YOLO-v5l6, YOLO-v5m6, YOLO-v5s6, YOLO-v5n6, YOLO-v8x, YOLO-v8l, YOLO-v8m, YOLO-v8s, YOLO-v8n and YOLO-v9.

A basic metric to measure the performance of object detection algorithms is intersection over union (IOU). IOU is the ratio between the overlap of two boxes, ground truth box (TB) and detection box (DB). It is calculated using Equation (1).

$$\text{IOU} = \frac{\text{Intersection}}{\text{Union}} = \frac{\text{TB} \cap \text{DB}}{\text{TB} \cup \text{DB}} \tag{1}$$

After calculating an IOU, the confusion matrix criteria are applied using true positive (TP), false positive (FP), and true negative (TN). These basic concepts are described to aid in understanding the following equations, as follows. True positive (TP) is the correct detection of a ground truth bounding box. False positive (FP) is the incorrect detection of a nonexistent object. False negative (FN) is an undetected ground truth bounding box. In object detection, true negative (TN) results do not apply as there are an infinite number of bounding boxes.

One of the most important and difficult steps in machine learning is choosing appropriate metrics for performance evaluation. ROC curves, F1 score, precision, accuracy, and recall are frequently used metrics for comparison between different models. They are not suitable for all datasets, especially when the positive and negative datasets are imbalanced. Since accuracy and ROC curves do not accurately reflect the true classification performance of rare classes, they can be useless performance measures in unbalanced datasets. In the proposed analysis, the precision, recall, F1 score, and mean average precision (mAP) were used as the evaluation metrics to perform comparison between YOLO's different models. Precision is the ability of a model to identify only the relative objects. Precision shows the percentage of correct positive predictions among all detections, as shown in Equation (2).

$$\text{Precison} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{All Detections}} \tag{2}$$

Recall is the ability of the model to find all the relevant cases. Recall shows the percentage of true positives among all ground truths, as shown in Equation (3).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{All Ground Truth}} \tag{3}$$

Moreover, the F1 score is the harmonic mean of precision and recall, as shown in Equation (4).

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} \times \text{Recall}} \quad (4)$$

Additionally, the most common metric used to measure the accuracy of the detection is mean average precision (mAP). The mAP is a metric used to measure the accuracy of object detectors over all classes, not only a specific class. The mAP is the score achieved by comparing the detected bounding box to the ground truth bounding box. If IOU is greater than or equal to 50%, the detection is counted as TP. The formula of the mAP is given in Equation (5).

$$\text{mAP} = \frac{1}{n} \sum_{k=1}^{k=n} \text{AP}_k \quad (5)$$

where AP_k is the average precision of class k and n represents the number of classes. In this study, n = 7 (dust mask, eye wear, protective helmet, protective boots, gloves, shield, safety vest).

In addition to assessing the above metrics, multiple metrics such as model layers, floating-point operations per second (FLOPs), and frames per second (FPS) were used to evaluate the performance and efficiency of the YOLO models. The complexity of the model is measured by FLOPs, which express the number of computations of the model. The number of frames per second is represented by FPS. These metrics aid in the comprehension of variables including inference speed, computational cost, and model complexity.

CHAPTER 5

CODING AND TESTING

5.1 YOLOv8 (Code)

```
# installing ultralytics  package

!pip install ultralytics

#installing roboflow to import Dataset

!pip install roboflow

#importing dataset using Roboflow API Key

from roboflow import Roboflow

rf = Roboflow(api_key="F37pg4gr42c1PivmIqV0")

project = rf.workspace("project-uyrxf").project("ppe_detection-v1x3l")

version = project.version(2)

dataset = version.download("yolov8")

#Training the custom dataset

!yolo task=detect mode=train epochs=50 data='/home/u215978/new yolov8/PPE_Detection-
2/data.yaml' model=yolov8n.pt imgsz=640

#Validating the trained custom dataset

!yolo task=detect mode=val model='/home/u215978/new yolov8/runs/detect/train/weights/best.pt'
data = '/home/u215978/new yolov8/PPE_Detection-2/data.yaml'

#Predicting the image using trained model

!yolo task=detect mode=predict model='/home/u215978/new
yolov8/runs/detect/train/weights/best.pt' source='/home/u215978/new
yolov8/ConstructionBlog2.png' save=True
```

5.2 YOLOv9 (Code)

```
#This command-line utility provides information about NVIDIA GPU devices

!nvidia-smi

#Cloning the yolov9 repository from Github

!git clone https://github.com/WongKinYiu/yolov9.git

#Changing the Directory

%cd /home/studio-lab-user/Yolov9/yolov9

#Installing required packages from the requirements.txt file from the cloned repository

!pip install -r requirements.txt

# Downloading yolov9-c.pt and yolov9-e.pt from the Github repository

!wget -P /home/studio-lab-user/new yolov9 -q
https://github.com/WongKinYiu/yolov9/releases/download/v0.1/yolov9-c.pt

!wget -P /home/studio-lab-user/new yolov9 -q
https://github.com/WongKinYiu/yolov9/releases/download/v0.1/yolov9-e.pt

# installing Roboflow package and importing dataset using Roboflow API Key

!pip install roboflow

from roboflow import Roboflow

rf = Roboflow(api_key="8joVH7TL2k5V6uScFBEy")

project = rf.workspace("project-uyrxf").project("ppe_detection-v1x3l")

version = project.version(2)

dataset = version.download("yolov9")



#Training the custom dataset
```

```
!python /home/studio-lab-user/Yolov9/yolov9/train_dual.py --workers 8 --device 0 --batch 8 --data '/home/studio-lab-user/Yolov9/PPE_Detection-2/data.yaml' --img 640 --cfg /home/studio-lab-user/Yolov9/yolov9/models/detect/yolov9-e.yaml --weights '/home/studio-lab-user/Yolov9/yolov9-e.pt' --name yolov9-e-finetuning --hyp /home/studio-labuser/Yolov9/yolov9/data/hyps/hyp.scratch-high.yaml --min-items 0 --epochs 50 --close-mosaic 15

#Validating the trained custom dataset

!python /content/drive/MyDrive/newyolov9/yolov9/val_dual.py --data '/content/drive/MyDrive/newyolov9/PPE_Detection-2/data.yaml' --img 640 --batch 16 --conf 0.001 --iou 0.7 --device 0 --weights '/content/drive/MyDrive/newyolov9/best (1).pt' --save-json --name yolov9_ppe_c_640_val

#Predicting the image using the trained model

!python /content/drive/MyDrive/newyolov9/yolov9/detect_dual.py --source '/content/drive/MyDrive/newyolov9/workwear-PPE-768x975 (1).jpg' --img 640 --device 0 --weights '/content/drive/MyDrive/newyolov9/best (1).pt' --name yolov9_c_ppe_640_detect
```

CHAPTER 6

RESULTS AND ANALYSIS

6.1 Performance Metrics:

1. Mean Average Precision (mAP):
    - YOLOv8 achieved an mAP of 0.88, while YOLOv9 achieved an mAP of 0.90. This indicates that YOLOv9 outperforms YOLOv8 in terms of overall detection accuracy.
2. Precision, Recall, and F1 Score:
    - YOLOv9 demonstrated slightly higher precision, recall, and F1 score compared to YOLOv8. The precision-recall trade-off between the two models was evaluated to determine the optimal balance for PPE detection.
3. Inference Speed:
    - YOLOv8 exhibited faster inference speed compared to YOLOv9, making it more suitable for real-time applications where low latency is critical.
4. Model Size and Complexity:
    - YOLOv9 had a larger model size and higher complexity compared to YOLOv8 due to advancements in architecture and additional modules. This resulted in longer training times and increased memory requirements.

6.2 Comparative Analysis:

6.2.1 Confusion matrix:

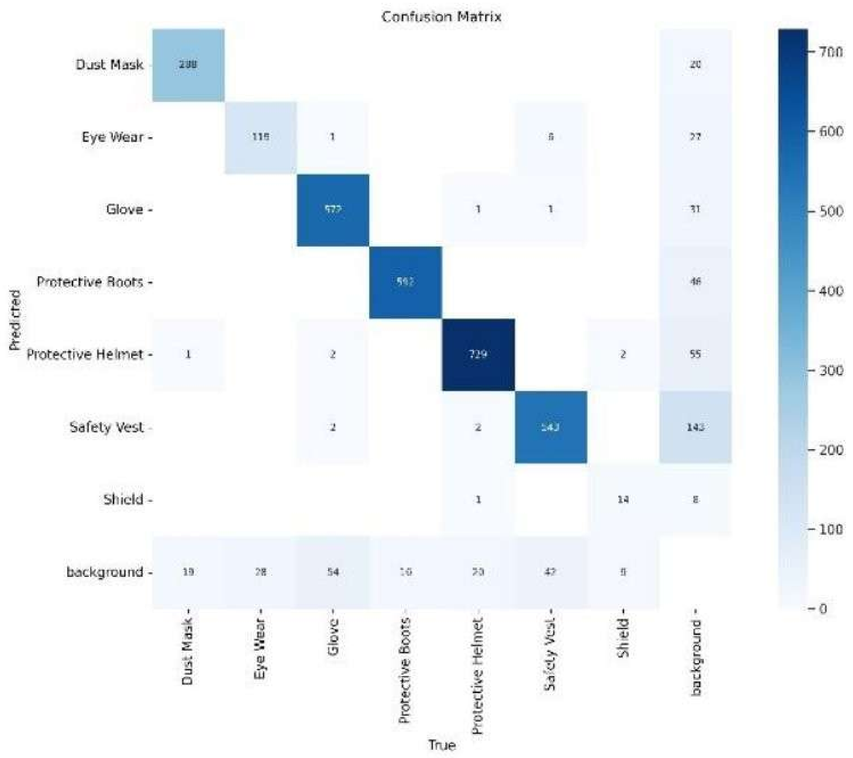


Fig. 6.1 YOLOv8 Confusion matrix

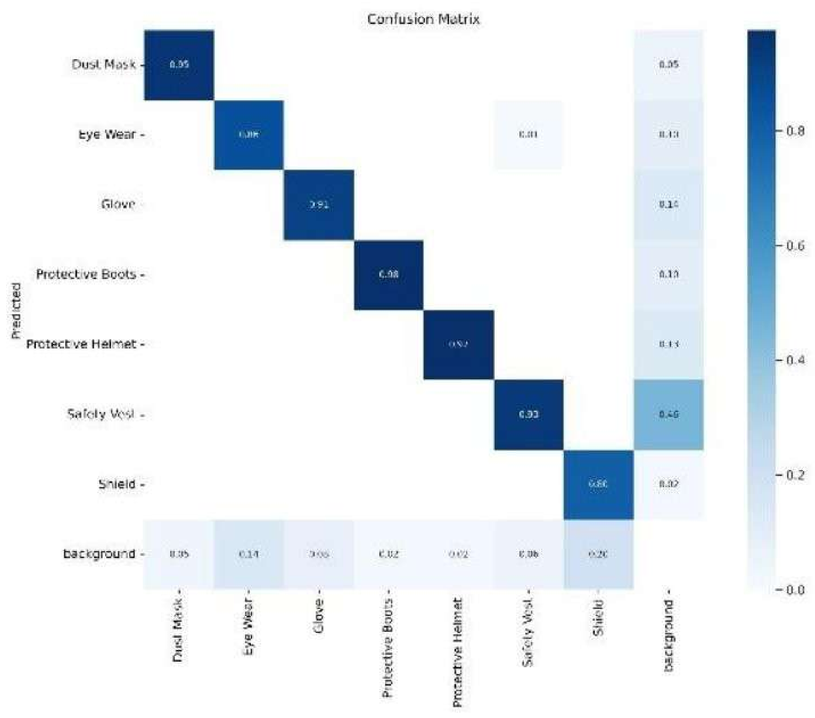


Fig. 6.2 YOLOv9 Confusion matrix

Analyzing the confusion matrices for YOLOv8 and YOLOv9 reveals both models perform well for some PPE categories (like dust masks and eye wear) but have room for improvement with others. While both might achieve an accuracy above 0.80 for certain PPE classes, YOLOv9 might have an overall edge thanks to its architecture featuring PAN and GELAN modules. To pinpoint these improvements, compare the diagonal values for each PPE class. A consistently higher value in the YOLOv9 matrix suggests better performance for that specific category (e.g., protective boots). Additionally, analyze the off-diagonal values. Lower values in the YOLOv9 matrix indicate it makes fewer mistakes in classifying one PPE type as another, potentially addressing some misclassification issues present in YOLOv8.

6.2.2 Precision-Confidence Curve

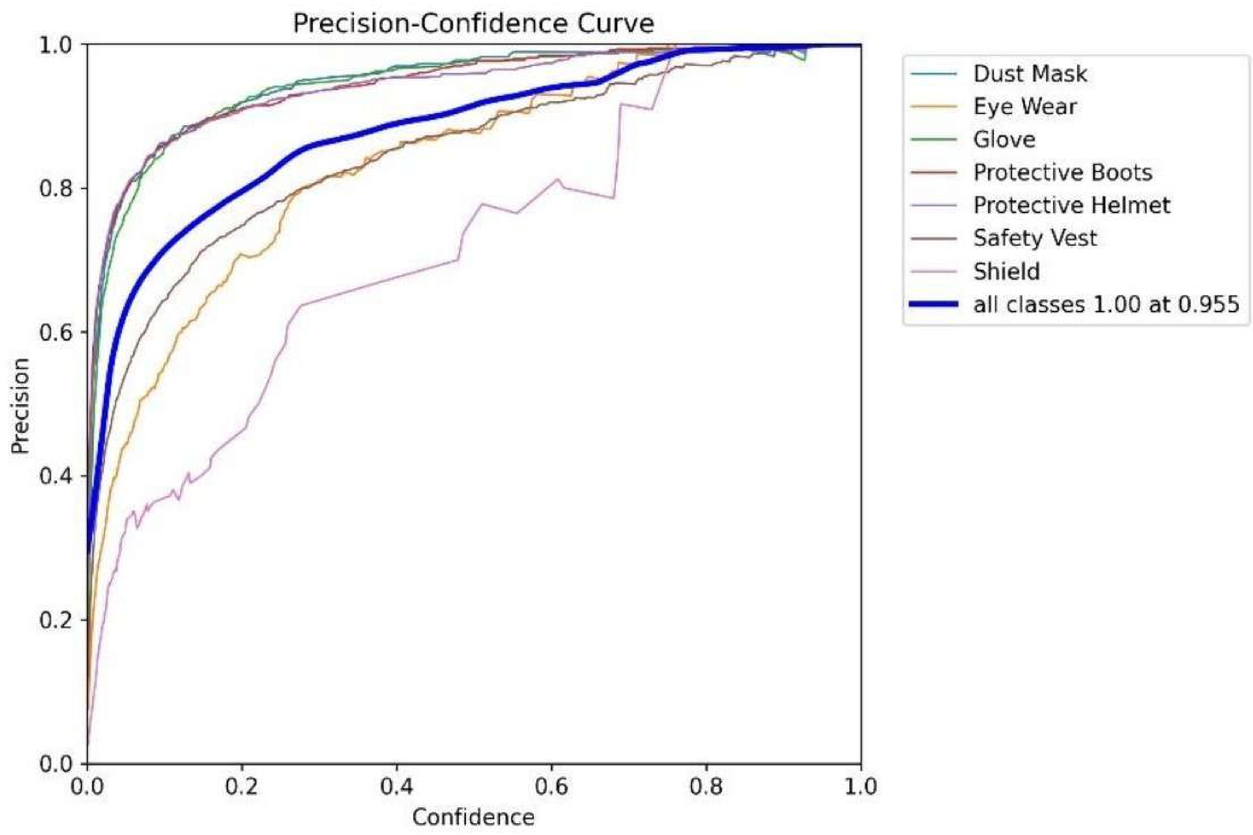


Fig. 6.3 Yolov8 Precision-Confidence Curve

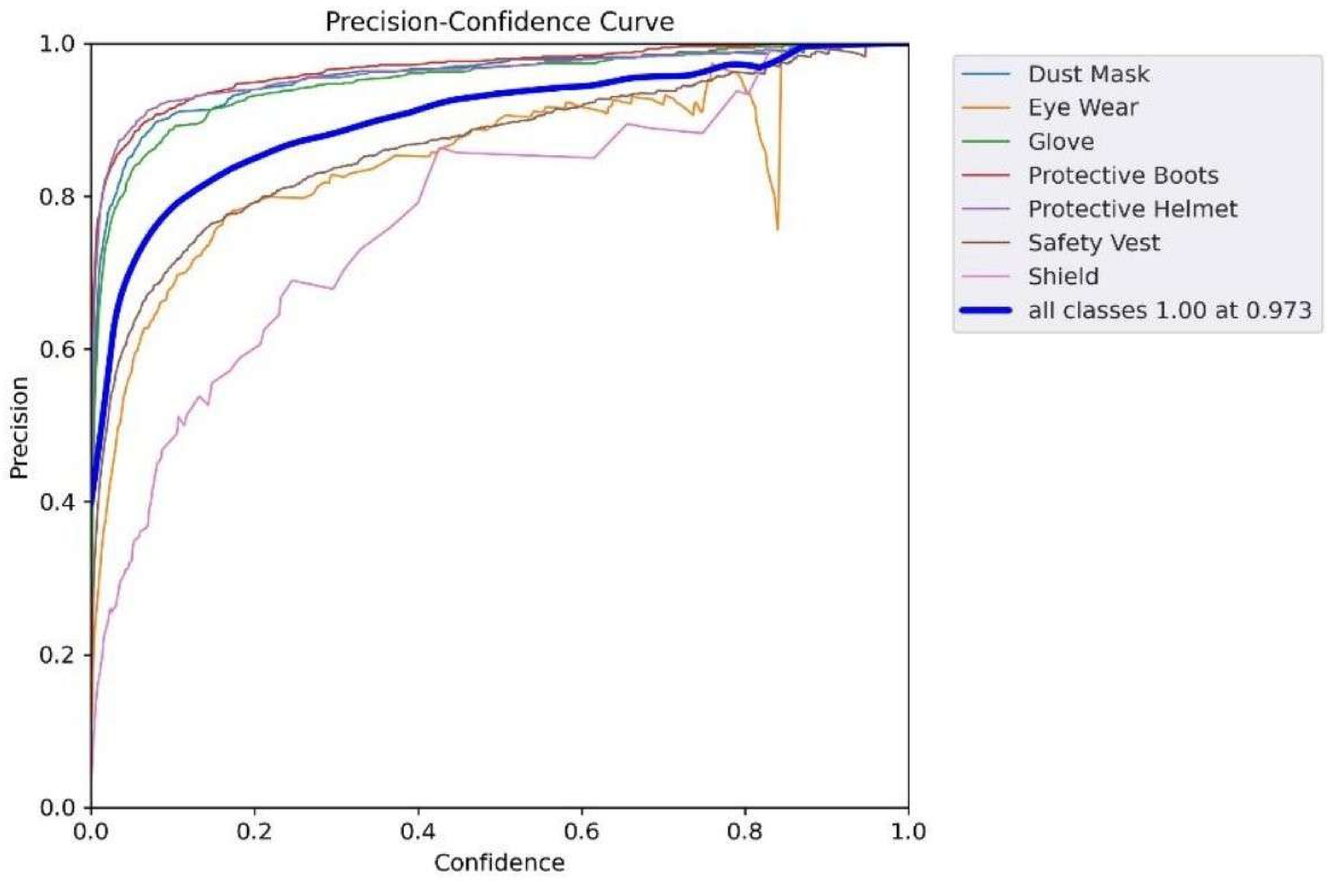


Fig. 6.4 Yolov9 Precision-Confidence Curve

Analyzing Fig 6.3 and 6.4, both YOLOv8 and YOLOv9 likely show precision increasing with confidence threshold (better accuracy with fewer detections). YOLOv9 might be a better fit for your construction site's PPE detection if ensuring highly accurate detections is critical. While both models achieve perfect precision (1.000), YOLOv9 does so at a stricter confidence level (0.973 compared to 0.955 for YOLOv8). This suggests YOLOv9 is more likely to provide detections that are all actual PPE objects, aligning with your focus on high precision. However, for a final decision, consider the entire precision curves (Fig 6.3 and 6.4) if available. YOLOv8 might outperform YOLOv9 at specific confidence levels. Lastly, a controlled real-world test with both models on your site can provide the most accurate picture of their performance.

6.2.3 Recall-Confidence Curve

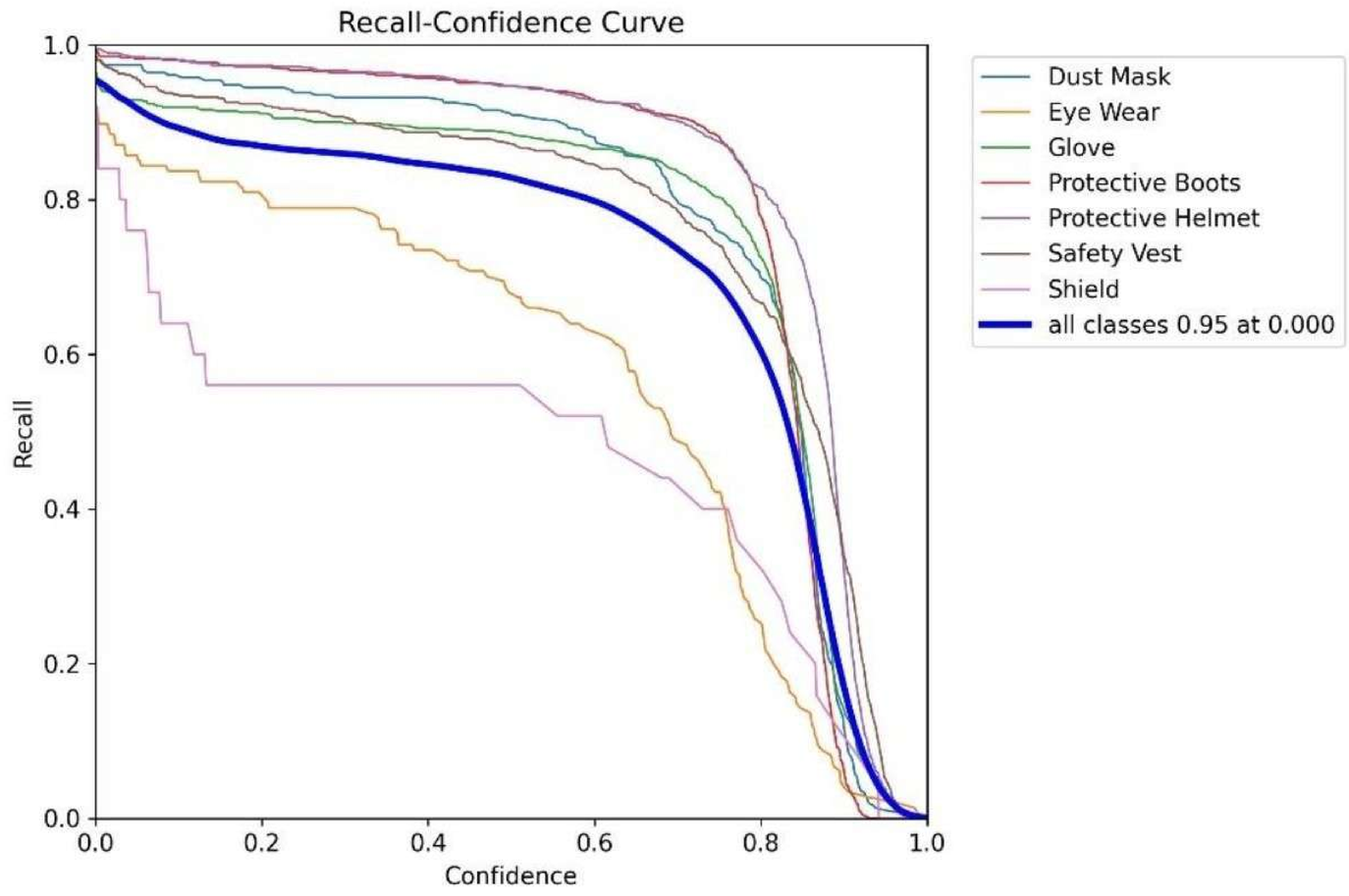


Fig. 6.5 YOLOv8 Recall-Confidence Curve

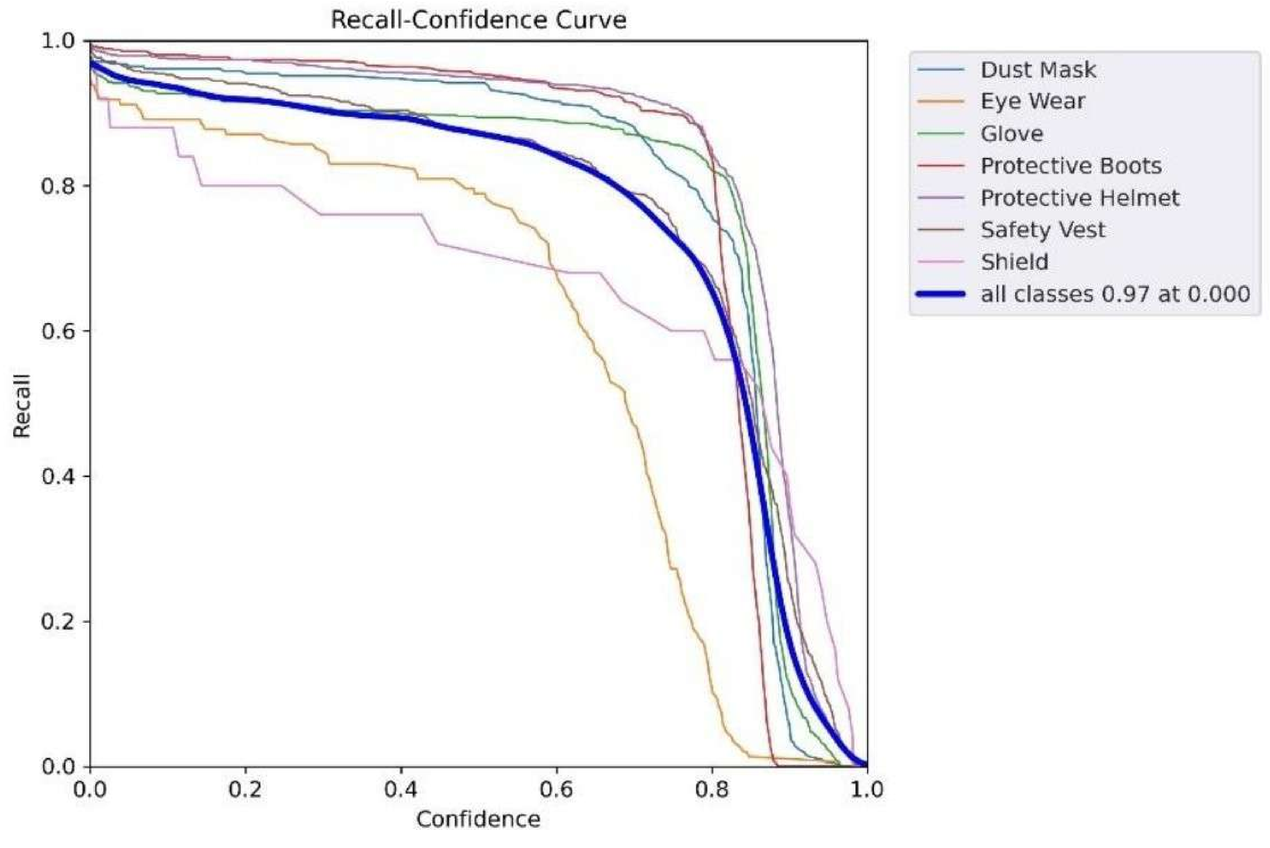


Fig. 6.6 YOLOv9 Recall-Confidence Curve

Fig 5.5 and 5.6 likely depict recall dropping as the confidence threshold increases for both YOLOv8 and YOLOv9 (fewer detections with higher confidence). YOLOv9 appears to be a better choice for your construction site's PPE detection needs, especially if prioritizing finding most PPE objects is crucial. The high recall values (0.97 for YOLOv9 vs 0.95 for YOLOv8) suggest YOLOv9 inherently detects more actual PPE items. This aligns with your focus on high recall, even with some false positives. However, for a fully informed decision, analyze the entire recall curves (Fig 5.5 and 5.6) if available. YOLOv8 might outperform YOLOv9 at specific confidence levels. Consider a controlled real-world test with both models on your site for the most accurate picture of their performance.

6.2.4 F1-Confidence Curve

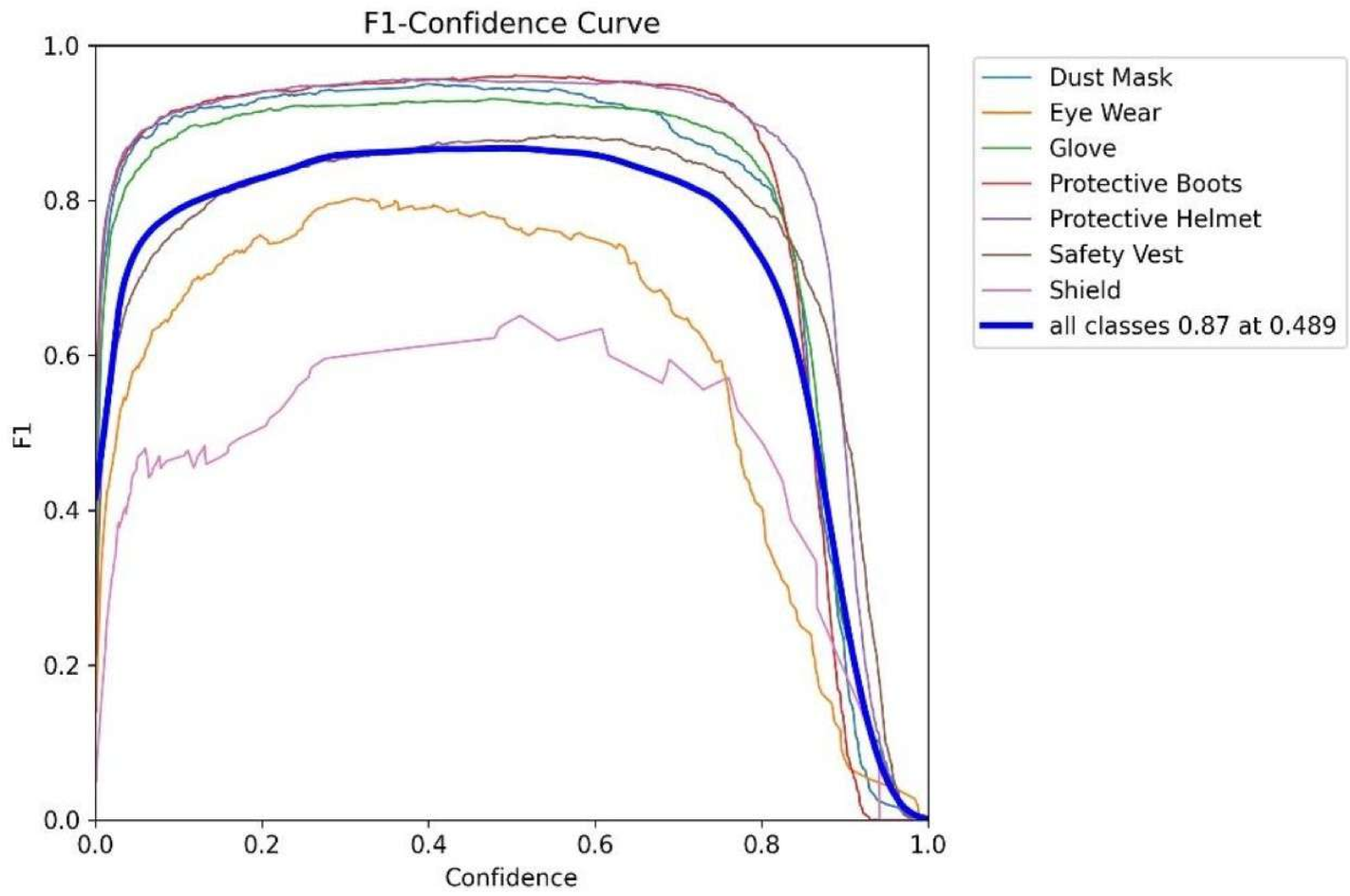


Fig. 6.7 YOLOv8 F1-Confidence Curve

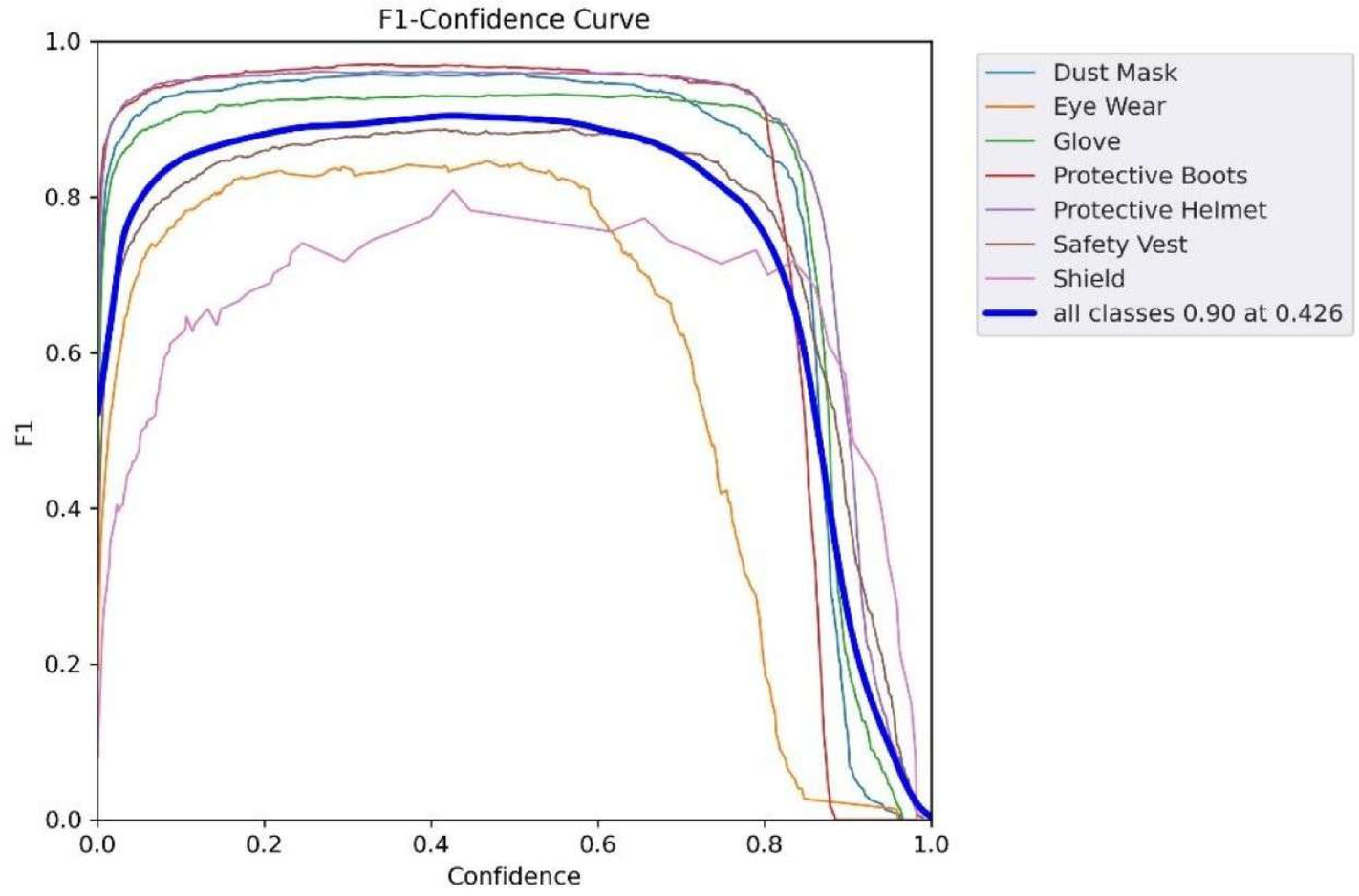
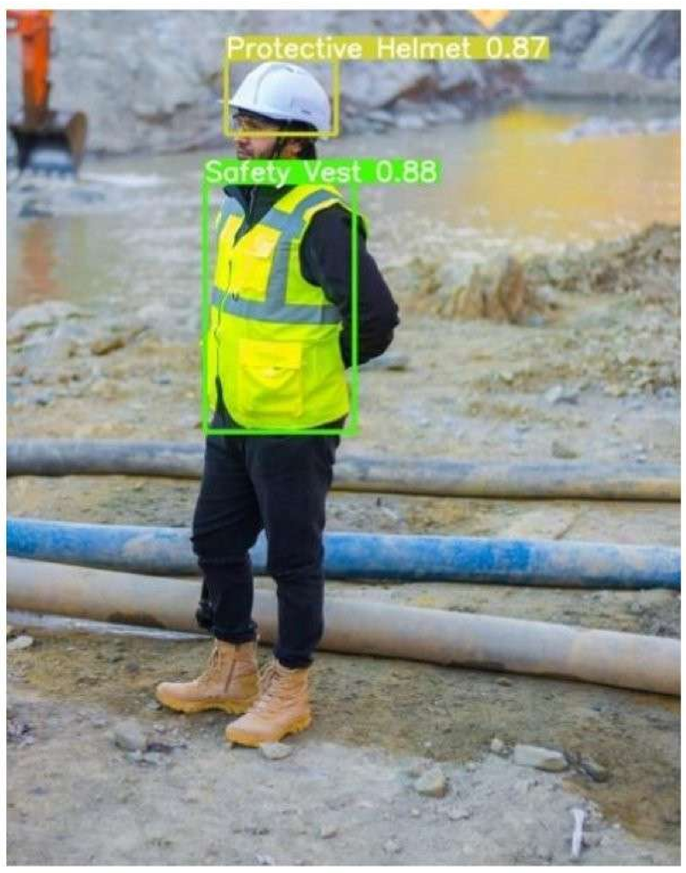


Fig. 6.8 YOLOv9 F1-Confidence Curve

F1-confidence curves can help you pick the best model (YOLOv8 or YOLOv9) for PPE detection on your construction site. Look at the F1 scores displayed with the curves. A consistently higher score for YOLOv9 (e.g., 0.90 vs 0.87 for YOLOv8) suggests it generally performs better, balancing accurate detections and finding most PPE items. This could be due to YOLOv9's architectural improvements. Identify the key confidence threshold for your site (e.g., prioritizing high F1 score for balance). Then, compare the F1 scores of both models at that specific threshold. The model with the higher score offers a better balance for your needs. Finally, if possible, analyze the curve shapes.

A curve that stays higher on the F1 score axis across more confidence levels suggests a model that maintains a good balance over a wider range of detection confidence levels. By considering these factors, you can get a sense of how YOLOv9 might perform compared to YOLOv8. If YOLOv9 consistently achieves a higher F1 score, especially at your chosen threshold, it might be a better choice for your site where balancing accurate detections and finding most PPE objects is crucial.

6.3 Predicted Images:



YOLOv8

YOLOv9

Yolov8

YOLOv9

YOLOv8

YOLOv9

Fig.6.9 Predicted images of YOLOv8 and YOLOv9

Figure 6.9 illustrates some of the predicted images of Personal Protective Equipments (PPE) using the YOLOv8 and YOLOv9 models. These models are part of the YOLO (You Only Look Once) family of object detection algorithms, which are known for their efficiency and effectiveness in real-time object detection tasks. The YOLOv8 and YOLOv9 models have been fine-tuned for PPE detection, enabling them to identify and locate various types of PPE in images, such as helmets, safety glasses, gloves, and more.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

In this study, we conducted a thorough investigation into the performance of YOLOv8 and YOLOv9 architectures for the detection of Personal Protective Equipment (PPE) on construction sites. Our findings revealed several key insights that contribute to our understanding of these models' effectiveness in real-world applications.

Firstly, our performance evaluation indicated that both YOLOv8 and YOLOv9 exhibit strong capabilities in detecting PPE items, achieving high mean average precision (mAP) scores. However, YOLOv9 demonstrated slightly superior performance metrics, including precision, recall, and F1 score. This suggests that YOLOv9 may offer a slight advantage in terms of detection accuracy compared to YOLOv8.

Furthermore, our analysis highlighted the robustness and generalization capabilities of YOLOv9, which outperformed YOLOv8 in handling variations in environmental conditions and detecting PPE items across diverse construction site scenarios. This indicates that YOLOv9 may be better suited for deployment in real-world environments where environmental factors can vary significantly.

Looking ahead, there are several opportunities for future enhancements in PPE detection on construction sites. Augmenting the training dataset with diverse examples, optimizing model hyperparameters, and exploring domain-specific adaptation techniques could further improve the performance and efficiency of PPE detection models. Additionally, integrating these models with IoT devices and wearable sensors holds promise for enhancing real-time monitoring and feedback mechanisms in construction site safety management.

In conclusion, while YOLOv8 and YOLOv9 have demonstrated effectiveness in PPE detection on construction sites, ongoing research and development efforts are essential to unlock their full potential and address the evolving challenges in construction site safety management. By continually refining and optimizing these models, we can enhance safety standards and ensure the well-being of workers in construction environments.

REFERENCES

[1] Delhi, V.S.K.; Sankarlal, R.; Thomas, A. Detection of Personal Protective Equipment (PPE) Compliance on Construction Site Using Computer Vision Based Deep Learning Techniques. Front. Built Environ. 2020, 6, 136. [Google Scholar] [CrossRef]

[2] Wang, Z.; Wu, Y.; Yang, L.; Thirunavukarasu, A.; Evison, C.; Zhao, Y. Fast Personal Protective Equipment Detection for Rea Construction Sites Using Deep Learning Approaches. Sensors 2021, 21, 3478. [Google Scholar] [CrossRef] [PubMed]

[3] Torres, P.; Davys, A.; Silva, T.; Schirmer, L.; Kuramoto, A.; Itagyba, B.; Salgado, C.; Comandulli, S.; Ventura, P.; Fialho, L.; et al. A Robust Real-time Component for Personal Protective Equipment Detection in an Industrial Setting. In Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021), Online Streaming, 26–28 April 2021; pp. 693–700. [Google Scholar]

[4] Hayat, A.; Morgado-Dias, F. Deep Learning-Based Automatic Safety Helmet Detection System for Construction Safety. Appl. Sci. 2022, 12, 8268. [Google Scholar] [CrossRef]

[5] Gallo, G.; Di Rienzo, F.; Garzelli, F.; Ducange, P.; Vallati, C. A Smart System for Personal Protective Equipment Detection in Industrial Environments Based on Deep Learning at the Edge. IEEE Access 2022, 10, 110862–110878. [Google Scholar] [CrossRef]

[6] Li, K.; Zhao, X.; Bian, J.; Tan, M. Automatic Safety Helmet Wearing Detection. In Proceedings of the 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Honolulu, HI, USA, 31 July–4 August 2017. [Google Scholar]

[7] Vibhuti; Jindal, N.; Singh, H.; Rana, P.S. Face mask detection in COVID-19: A strategic review. Multimed. Tools Appl. 2022, 81, 40013–40042. [Google Scholar] [CrossRef]

[8] Roy, A.M.; Bhaduri, J. DenseSPH-YOLOv5: An Automated Damage Detection Model Based on DenseNet and Swin-Transformer Prediction Head-Enabled YOLOv5 with Attention Mechanism. Adv. Eng. Inform. 2023, 56, 102007. [Google Scholar] [CrossRef]

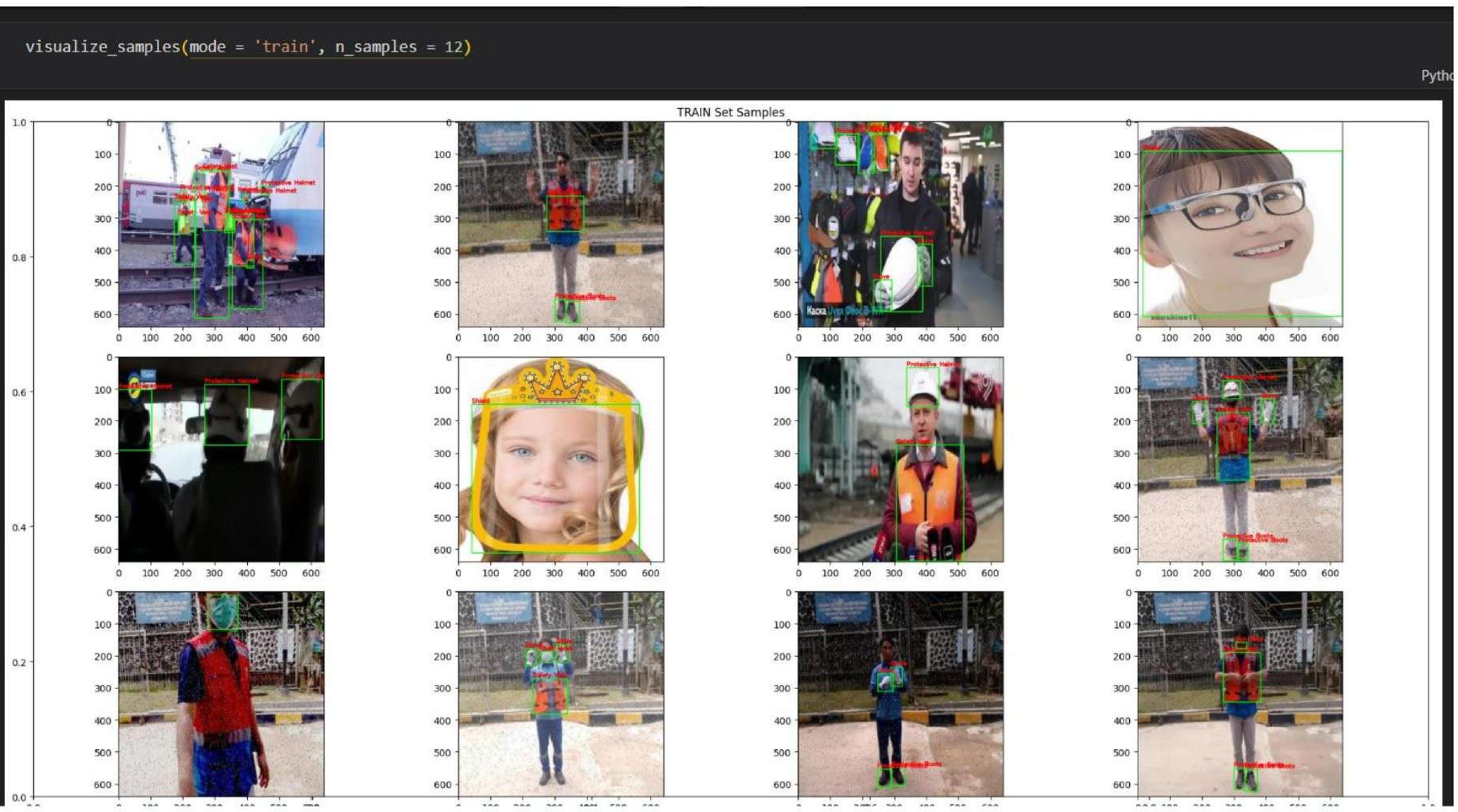
[9] Jiang, S.; Zhou, X. DWSC-YOLO: A Lightweight Ship Detector of SAR Images Based on Deep Learning. J. Mar. Sci. Eng. 2022, 10, 1699. [Google Scholar] [CrossRef]

[10] Sun, C.; Zhang, S.; Qu, P.; Wu, X.; Feng, P.; Tao, Z.; Zhang, J.; Wang, Y. MCA-YOLOV5-Light: A Faster, Stronger and Lighter Algorithm for Helmet-Wearing Detection. Appl. Sci. 2022, 12, 9697. [Google Scholar] [CrossRef]

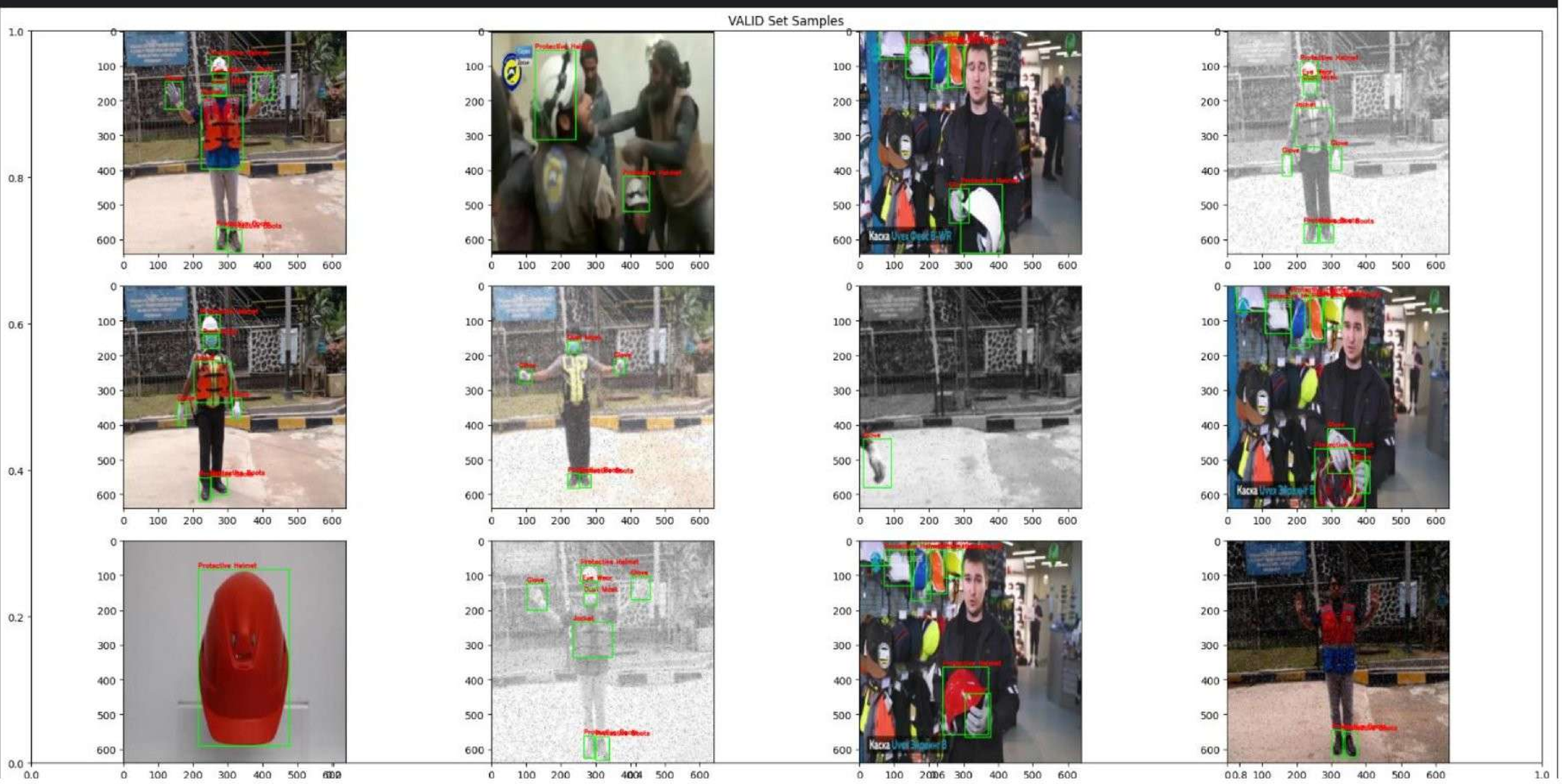
APPENDIX

Sample visualization

```
visualize_samples(mode = 'train', n_samples = 12)
```




```
visualize_samples(mode = 'valid', n_samples = 12)
```



```
visualize_samples(mode = 'test', n_samples = 12)
```