# AI Project Algorithm Report


Session: 2021 – 2025

**Submitted To:**

Mr. Samyan Qayyum Wahla

**Submitted By:**

Hammad Younas 2021-CS-20

Department of Computer Science

**University of Engineering and Technology Lahore Pakistan**

# Contents

# safety helmet and no helmet detection YOLOv5

## 1 Introduction

The Safety Helmet and No Helmet Detection using YOLOv5 project is aimed at enhancing safety measures by automating the identification of individuals wearing safety helmets in various scenarios. The project utilizes the YOLOv5 object detection architecture, known for its efficiency and accuracy in detecting multiple objects within images and videos in real-time.

## 2 Background

Workplace safety is a critical concern across industries, particularly in environments where hazards pose significant risks to workers' well-being. Among the essential safety protocols, the use of safety helmets stands as a fundamental measure to mitigate head injuries in high-risk settings such as construction sites, industrial zones, and manufacturing plants. Traditional safety compliance monitoring relies heavily on manual inspections, which are time-consuming and prone to human error, making it challenging to ensure comprehensive adherence to safety regulations, especially in large-scale operations.

With advancements in technology, there's a growing emphasis on automating safety measures. Object detection systems empowered by cutting-edge machine learning models, such as YOLOv5, offer a promising avenue to automate the identification of individuals wearing safety helmets and those without proper headgear. The Safety Helmet and No Helmet Detection using YOLOv5 project aims to address these challenges by leveraging this state-of-the-art technology to create an automated system capable of swiftly and accurately detecting safety helmets in images or video streams.

YOLOv5, known for its efficiency and accuracy in real-time object detection, is particularly well-suited for this purpose. Its capability to process visual data rapidly while precisely recognizing safety helmets and individuals not compliant with safety protocols makes it an ideal solution for enhancing safety measures. Implementing such an automated system not only ensures strict adherence to safety regulations but also significantly reduces the risks of head injuries, fosters a stronger safety culture in workplaces, and streamlines operations by minimizing the need for manual supervision.

## 3 Objective

### 3.1 Safety Enhancement

The primary goal is to enhance safety measures by automating the detection of individuals wearing safety helmets in environments where it is mandatory.

### 3.2 Efficiency in Object Detection

Leveraging YOLOv5's capabilities, the project aims to achieve high accuracy and real-time detection of safety helmets and individuals without helmets.

### 3.3 Automation

By automating the detection process, the system reduces the need for manual inspection, thereby saving time and improving overall efficiency.

## 4 Key Features

### 4.1 YOLOv5 Object Detection

The project uses YOLOv5, a state-of-the-art object detection model, to precisely locate and classify safety helmets and individuals without helmets in images or video frames.

### 4.2 Versatility

The system is versatile and can be deployed across various industries and scenarios where safety helmet compliance is essential, including construction sites, industrial facilities, and more.

### 4.3 Real-time Detection

The model's architecture allows for near real-time detection, ensuring prompt identification and immediate action if safety regulations are violated.

## 5 Code Appendix

A section containing key excerpts of the Python code used in the project for reference. This includes data loading, preprocessing steps, model training, and evaluation.

### 5.1 Importing libraries

```
import cv2
import torch
import numpy as np
```

### 5.2 Loading the Model

```
path = 'E:\\5th semester\\artifical intelligence\\helmet detection demo
\\safety-helmet-and-no-helmet-detection-using-YOLOv5\\yolov5safetyhelmet-main
\\yolov5safetyhelmet-main\\yolov5safetyhelmet-main\\best.pt'

model = torch.hub.load('ultralytics/yolov5', 'custom',path, force_reload=True)
```

# 6 Explanation of Safety Helmet Detection Code

## 6.1 Initializing the Video Capture:

The code initializes the video capture object `cap` using OpenCV's `VideoCapture` function to access the default camera (index 0).

```
cap = cv2.VideoCapture(0)
```

## 6.2 Initializing Variables:

A counter `count` is initialized to keep track of the number of frames processed.

```
count = 0
```

## 6.3 Processing Frames in a Loop:

This loop continuously captures frames from the camera until there are no more frames (`ret` becomes False). The loop increments the `count` variable for each frame processed.

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
    count += 1
```

## 6.4 Skipping Frames:

This conditional statement skips frames to reduce the processing load. It continues the loop only if the frame count is not a multiple of 3. It effectively processes every third frame.

```
    if count % 3 != 0:
        continue
```

## 6.5 Resizing the Frame:

The captured frame is resized to a specific dimension (1020x600) using OpenCV's `resize` function.

```
    frame = cv2.resize(frame, (1020, 600))
```

## 6.6 Processing the Frame with a Model:

This section represents the inference part where a pre-trained model is used to analyze the resized frame. The `model()` function processes the frame, and the inference results are rendered onto the frame. The `results.render()` method extracts the rendered frame from the model's output.

```
    results = model(frame)
    frame = np.squeeze(results.render())
```

### 6.7 Displaying the Processed Frame:

The processed frame is displayed in a window titled ”FRAME” using `cv2.imshow()`. The `cv2.waitKey(1)` waits for a key event for 1 millisecond, and if the key pressed is the ”Esc” key (0xFF == 27 in ASCII), the loop breaks, terminating the video capture.

```
    cv2.imshow("FRAME", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break
```

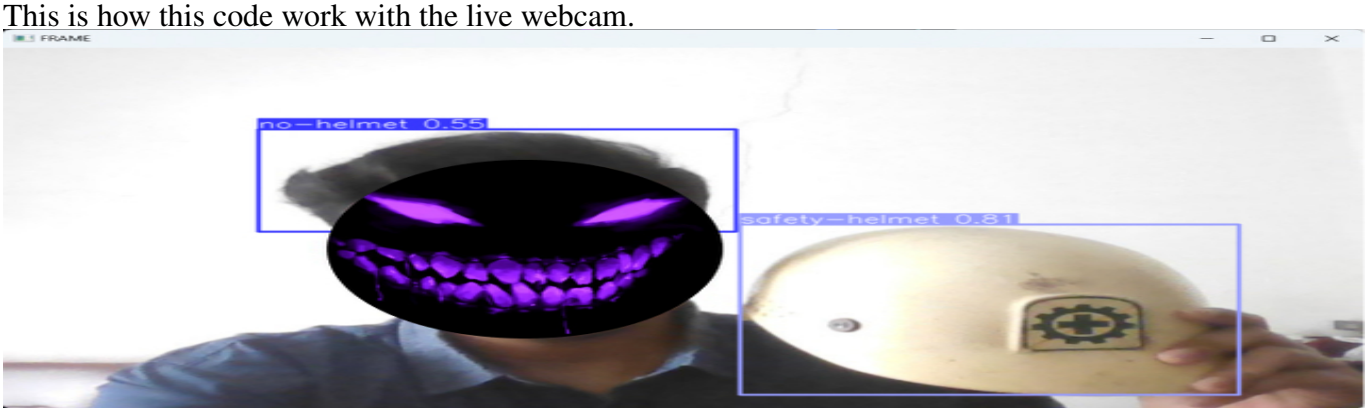
### 6.8 Releasing Resources:

Finally, the video capture object (`cap`) is released, and OpenCV windows are destroyed, releasing the system resources.

```
cap.release()
cv2.destroyAllWindows()
```

### 6.9 Processing:

This is how this code work with the live webcam.




## 7 Conclusion

In conclusion, the presented code demonstrates a systematic approach to safety helmet detection utilizing OpenCV and machine learning models. Through the initialization of video capture, processing frames with a model, and displaying processed frames, this code represents a foundational framework for real-time safety compliance monitoring. The implementation of such systems holds significant potential in enhancing workplace safety, automating compliance checks, and reducing the reliance on manual supervision. Further advancements and optimizations in machine learning models and computer vision techniques can foster even more robust and accurate safety monitoring systems in various industrial and safety-critical environments.