SafeZone : Real-time Video Analytics for Industrial Safety

TEAM NAME: The Tech Titians

TEAM MEMBERS:

M. Ragavaa
S. Rathesh
Y. Dharineesh
S. Bhuvanraj.
[Sona College of Technology, III Year AI&Ds Department]

Problem Statement:

To develop a system that can automatically and accurately detect whether or not workers are wearing personal protective equipment (PPE). PPE is essential for protecting workers from hazards in the workplace, and non-compliance with PPE requirements can lead to serious injuries or death.

The problem statement for ***SafeZone: Real-time Video Analytics for Industrial Safety*** can be broken down into the following sub-problems:

- **Object detection**: The system must be able to detect people and PPE items in images or video. This is a challenging problem due to the variability in the appearance of PPE items, the occlusion of PPE items by other objects, and the cluttered background in many workplace environments.

- **Classification**: The system must be able to classify each PPE item as either present or not present. This is also a challenging problem due to the variability in the appearance of PPE items and the possibility of PPE items being partially obscured.
- **Real-time operation**: The system must be able to operate in real time, so that it can be used to monitor workers in a dynamic workplace environment.

MILESTONE 1 : Pre-Requisite

Tools Used:

1. ultralytics - For YOLO
2. YOLOv8 - For dataset and model training
3. CVzone – To connect with videos and webcam
4. Programming Language – Python 11
5. Environment – PyCharm

Data Collection:

- We have collected the dataset from Roboflow which is a untrained dataset and by using YOLO (Version 8) we are training those dataset.

Explanation Video
https://drive.google.com/file/d/109oWMxce_kqNcficUxwPHAG_KHCbyg87/view?usp=sharing

Custom Data Training:

- As the accuracy of the readily available dataset is low, we have collected the images and trained the images for developing the model.
- Custom data training in PPE is the process of training a machine learning model to detect personal protective

equipment (PPE) in images or videos. This can be done by collecting a dataset of images or videos that contain PPE, and then annotating each image or video with the location and type of PPE. The annotated dataset is then used to train the machine learning model.

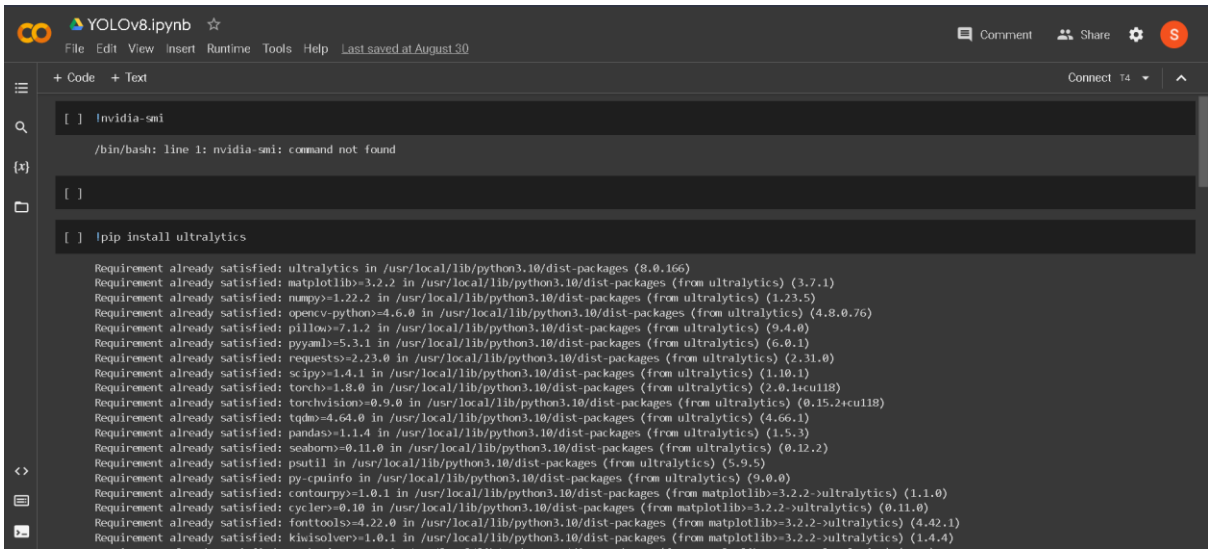
Explanation Video

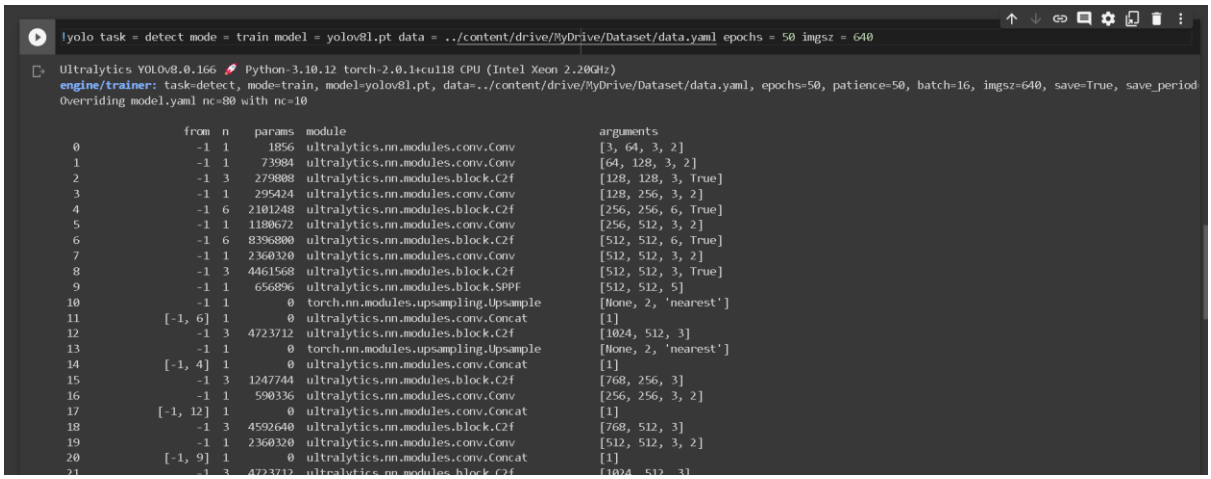
https://drive.google.com/file/d/1jhBb8in9uQbwBZVW3IeWOec8v9xNKu_H/view?usp=sharing

MILESTONE 2 : BUILD SOLUTION

2.1 – CUSTOM DATA TRAINING:

- For Custom Data Training, we have chosen Google Colab as the environment because we doesn't have any delicated graphics card in my laptop. Google Colab provides a virtual GPU to some extend .
- We are dividing the 10 groups in the model
  Class Names are:
  ['bare_arm', 'boot', 'glasses', 'glove', 'hard_hat', 'mask', 'no_glove', 'no_hard_hat', 'no_mask', 'no_vest', 'person', 'vest']

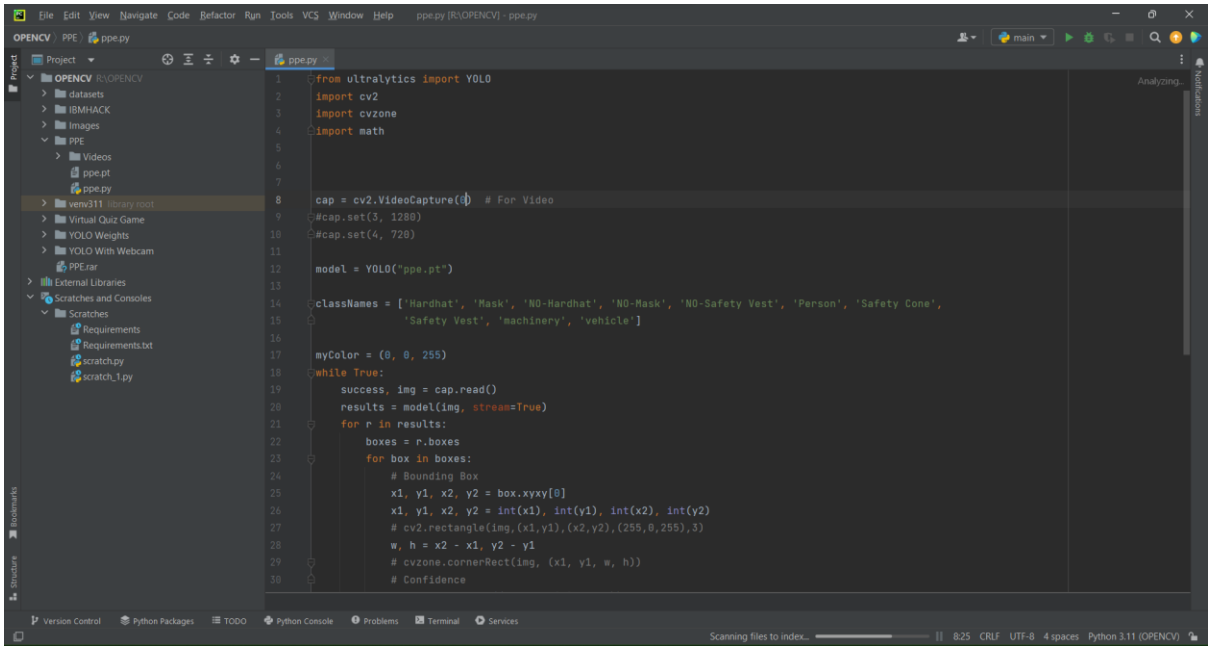




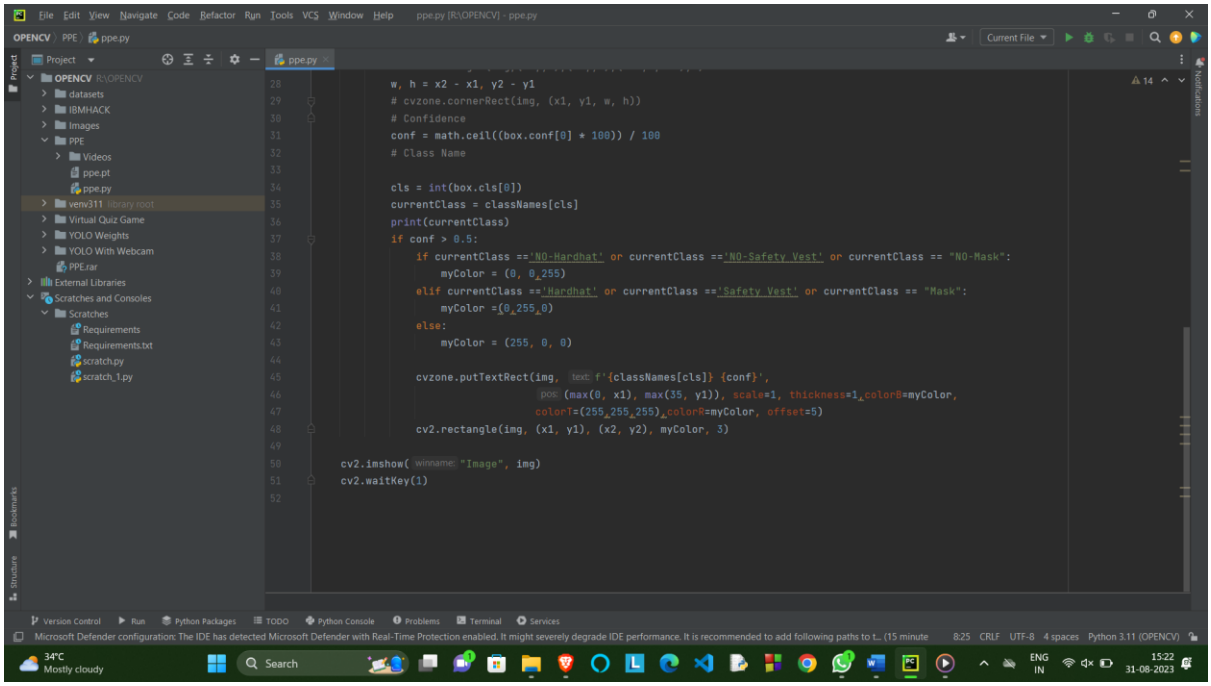
- After the completion of all the epochs, Download the Trained Weights from Colab as ”ppe.pt”.

2.2 MODEL TRAINING:

- After training the dataset we take weights from it.
- We train models with the help of opencv library.
- Explanation Video https://drive.google.com/file/d/165Gul19dcheMsKU3Uf1X-AZ5A0Haonrp/view?usp=sharing



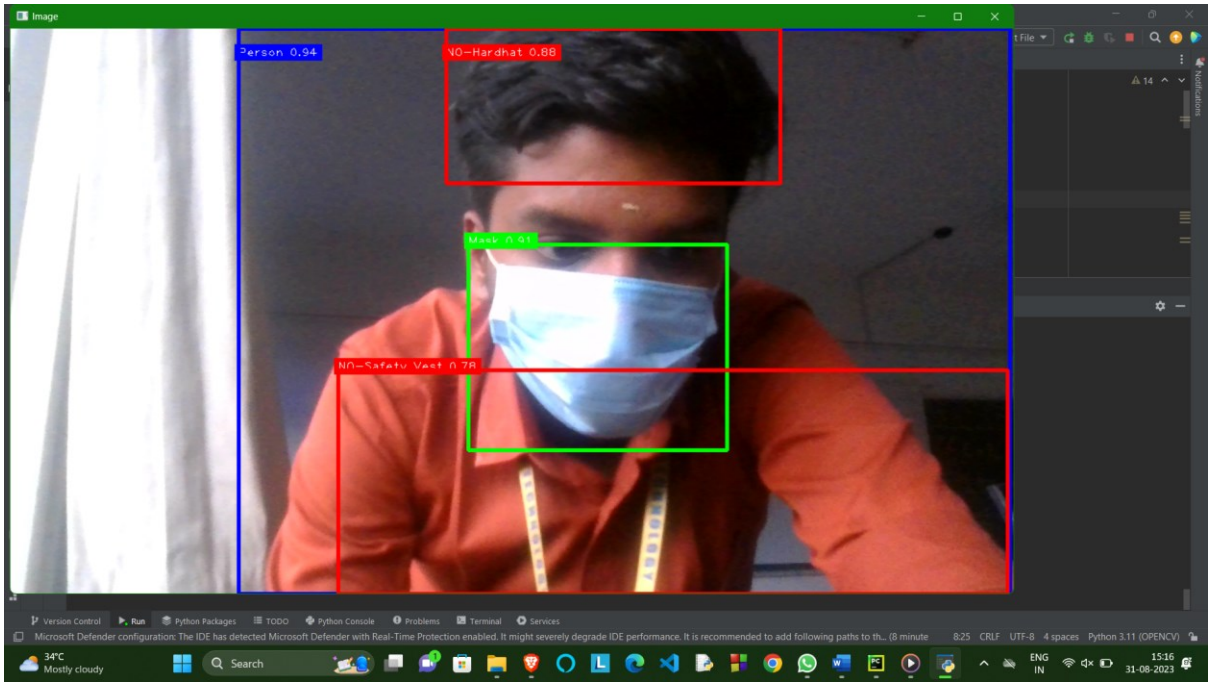
```python
from ultralytics import YOLO
import cv2
import cvzone
import math



cap = cv2.VideoCapture(0)  # For Video
#cap.set(3, 1280)
#cap.set(4, 720)

model = YOLO("ppe.pt")

classNames = ['Hardhat', 'Mask', 'NO-Hardhat', 'NO-Mask', 'NO-Safety Vest', 'Person', 'Safety Cone',
              'Safety Vest', 'machinery', 'vehicle']

myColor = (0, 0, 255)
while True:
    success, img = cap.read()
    results = model(img, stream=True)
    for r in results:
        boxes = r.boxes
        for box in boxes:
            # Bounding Box
            x1, y1, x2, y2 = box.xyxy[0]
            x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
            # cv2.rectangle(img,(x1,y1),(x2,y2),(255,0,255),3)
            w, h = x2 - x1, y2 - y1
            # cvzone.cornerRect(img, (x1, y1, w, h))
            # Confidence
```



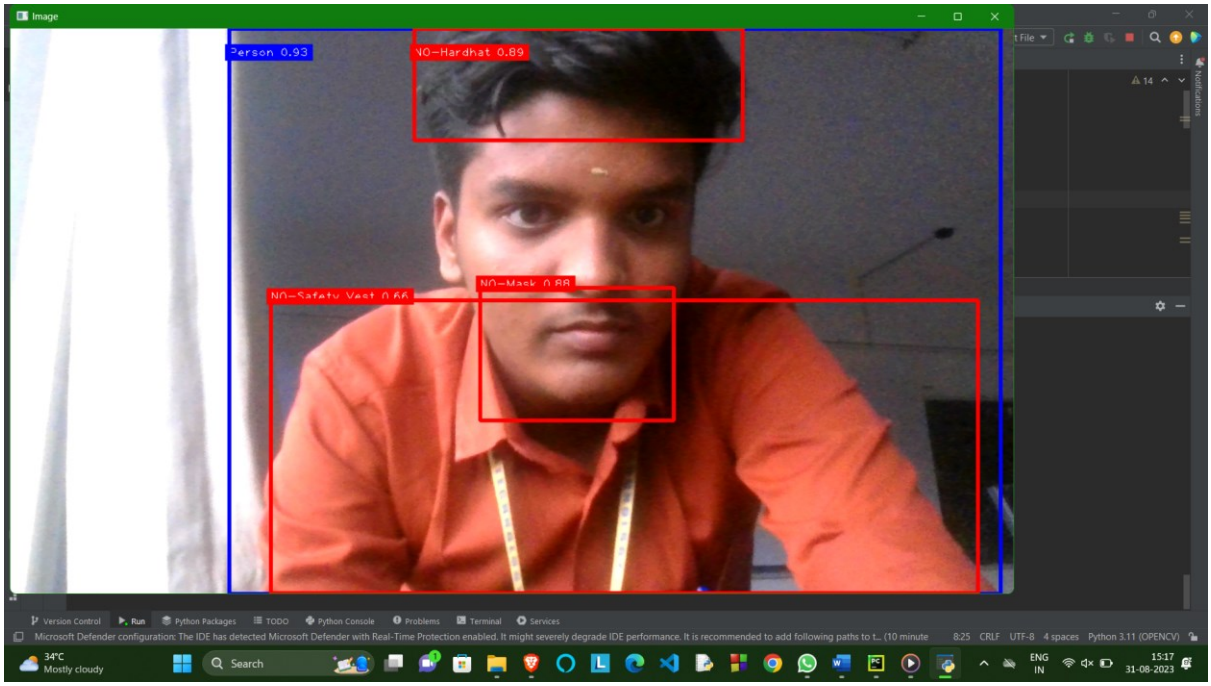
```python
            w, h = x2 - x1, y2 - y1
            # cvzone.cornerRect(img, (x1, y1, w, h))
            # Confidence
            conf = math.ceil((box.conf[0] * 100)) / 100
            # Class Name

            cls = int(box.cls[0])
            currentClass = classNames[cls]
            print(currentClass)
            if conf > 0.5:
                if currentClass =='NO-Hardhat' or currentClass =='NO-Safety Vest' or currentClass == "NO-Mask":
                    myColor = (0, 0,255)
                elif currentClass =='Hardhat' or currentClass =='Safety Vest' or currentClass == "Mask":
                    myColor =(0,255,0)
                else:
                    myColor = (255, 0, 0)

                cvzone.putTextRect(img,  text: f'{classNames[cls]} {conf}',
                                   pos: (max(0, x1), max(35, y1)), scale=1, thickness=1,colorB=myColor,
                                   colorT=(255,255,255),colorR=myColor, offset=5)
                cv2.rectangle(img, (x1, y1), (x2, y2), myColor, 3)

    cv2.imshow( winname: "Image", img)
    cv2.waitKey(1)
```

OUTPUT:

1-With Mask



2-Without Mask



NOTE:

- I have enclosed all the documents related to this project in the GitHub repository, Kindly refer it.