GuardianEye: Yolo-Based Smart Helmet Detection System For Enhanced Safety In Real-Time

Introduction:

"GuardianEye" is a cutting-edge project introducing a YOLO-based smart helmet detection system designed to bolster safety measures in real-time environments. By integrating YOLO (You Only Look Once), a powerful object detection algorithm, into helmets, this system actively identifies and alerts users to potential hazards, enhancing safety across various scenarios.

Scenario 1:

Construction sites are notorious for their inherent risks, including falling objects and machinery accidents. With Guardian Eye installed on their helmets, workers benefit from real-time detection of hazardous situations. The system swiftly identifies falling debris or approaching machinery and issues immediate alerts, enabling workers to take evasive action, thereby mitigating potential injuries.

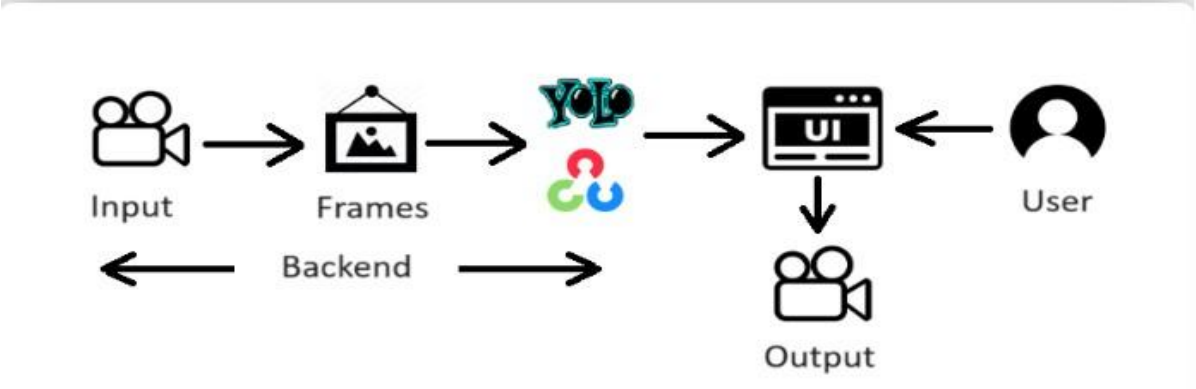
Scenario 2:

Motorcyclists navigate roads fraught with dangers, from erratic drivers to unexpected obstacles. GuardianEye revolutionizes motorcycle safety by equipping riders with helmets featuring its advanced detection technology. As bikers traverse busy streets, the system detects nearby vehicles, pedestrians, and road hazards, issuing timely alerts to enhance situational awareness and reduce the likelihood of collisions.

Scenario 3:

Emergency responders face perilous conditions while executing their duties, often operating in low-visibility environments or amidst chaos. GuardianEye provides indispensable support by integrating its detection capabilities into its helmets. Whether navigating through smoke-filled buildings or responding to traffic accidents, responders receive real-time alerts about potential dangers, empowering them to make informed decisions and safeguard lives more effectively.

Technical Architecture:



Prior Knowledge:

You must have prior knowledge of the following topics to complete this project.

- YOLO v5- https://www.youtube.com/watch?v=ag3DLKsl2vk
- OpenCV - https://www.youtube.com/watch?v=WQeoO7MI0Bs
- Flask - https://www.youtube.com/watch?v=lj4I_CvBnt0

Prerequisites:

To complete this project, you must require the following software, concepts, and packages.

1. IDE Installation:

Spyder/ PyCharm IDE is Ideal to complete this project

To install Spyder IDE, please refer to Spyder IDE Installation Steps

To install PyCharm IDE, please refer to the PyCharm IDE Installation steps

2. Python Packages

If you are using Anaconda Navigator, follow the below steps to download the required packages:

Open the Anaconda prompt and create a virtual environment.

- Type “conda with base environment python=3.11”.
- Type "pip install opencv-python==4.7.0.68” and click enter.
- Type "pip install flask” and click enter.

Project Objectives:

- Know fundamental concepts and techniques used for computer vision.

  Gain a broad understanding of the YOLO.

  Gain knowledge of OpenCV.

Project Flow:

- The user interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once the model analyses the input the summary is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

Create App.py file

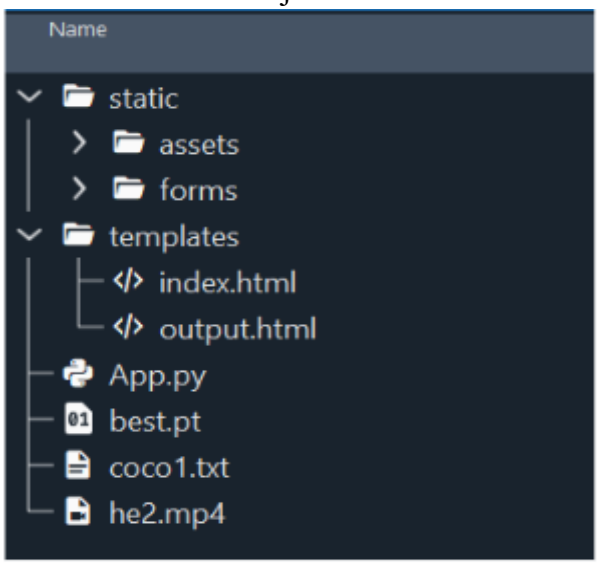
- Import the required libraries
- Load Pre-Trained weights
- Load Pre-Trained model with Yolov8

Application building

- Building HTML page
- Build Python code
- Run the application

Project Structure:

 Create a Project folder which contains files as shown below

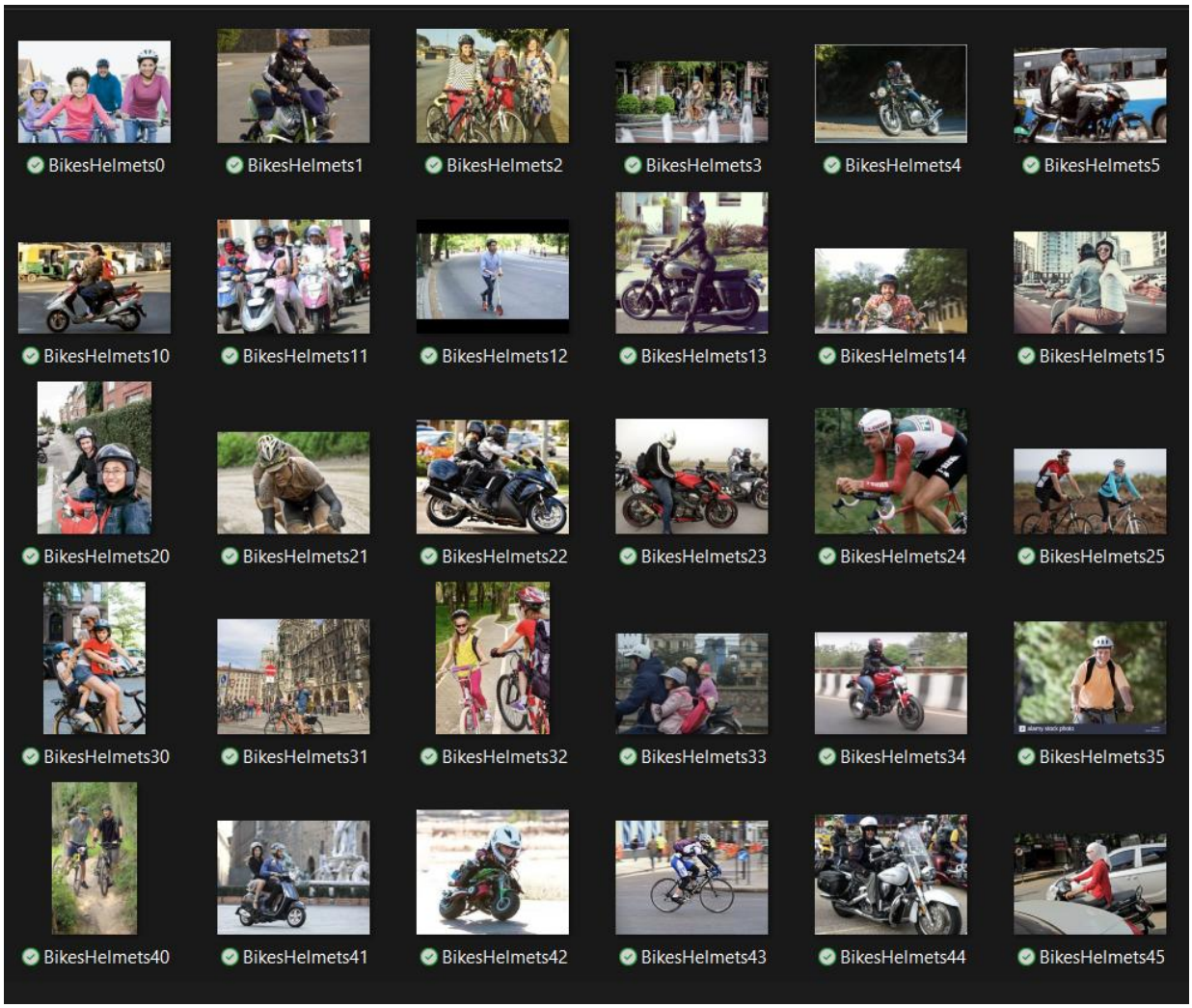


The Template folder contains HTML pages. The App.py contains the Flask code for real-time helmet detection and social distance verification. This innovative solution seamlessly integrates computer vision techniques to enhance safety protocols in various settings.

Milestone 1: Collection of Data

Dataset has three classes swimming, drowning, out of water. **Download the Dataset-**

https://www.kaggle.com/datasets/andrewmvd/helmet-detection



Sample data:

Milestone 2: Image Pre-processing

In this milestone, we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although performing some geometric transformations of images like rotation, scaling, translation, etc.

Activity 1: Import the required libraries

We need to download yoloV5 from the github

```
import locale
def getpreferredencoding(do_set=True):
    return "UTF-8"
locale.getpreferredencoding = getpreferredencoding
!wget https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5s.pt
```

```
--2024-12-03 13:16:42--  https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5s.pt
Resolving github.com (github.com)... 140.82.114.4
Connecting to github.com (github.com)|140.82.114.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/eab38592-7168-4731-bdff-ad5ede20
--2024-12-03 13:16:42--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/264818686/eab38592-7168-473
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14698491 (14M) [application/octet-stream]
Saving to: 'yolov5s.pt.28'

yolov5s.pt.28       100%[===================>]  14.02M  51.5MB/s    in 0.3s

2024-12-03 13:16:43 (51.5 MB/s) - 'yolov5s.pt.28' saved [14698491/14698491]
```

Activity 2: Loading the Dataset

Download the dataset from the kaggle using CLI it will download zip file and unzip the file to get train, valid, test folders.

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="aGEhgGkUV19vTP7EKVGr")
project = rf.workspace("helmet-detection-system-for-enhanced-safety-in-realtime-9kmwx").project("hard-hat-sample-tpt0x")
version = project.version(1)
dataset = version.download("yolov5")
```

```
Collecting roboflow
  Downloading roboflow-1.1.49-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from roboflow) (2024.8.30)
Collecting idna==3.7 (from roboflow)
  Downloading idna-3.7-py3-none-any.whl.metadata (9.9 kB)
Requirement already satisfied: cycler in /usr/local/lib/python3.10/dist-packages (from roboflow) (0.12.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.4.7)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from roboflow) (3.8.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.26.4)
Requirement already satisfied: opencv-python-headless==4.10.0.84 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.10.0.84)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from roboflow) (11.0.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.8.2)
Collecting python-dotenv (from roboflow)
  Downloading python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.32.3)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.16.0)
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.2.3)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.66.6)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (6.0.2)
Requirement already satisfied: requests-toolbelt in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.0.0)
Collecting filetype (from roboflow)
  Downloading filetype-1.2.0-py2.py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (1.3.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (4.55.0)
```

Pre-processing:

```
results[0].plot()
```

```
image_path = "/content/drive/MyDrive/helmet/runs/detect/predict/bus.jpg"
results = model(image_path)
result = results[0]
boxes = result.boxes.xyxy
labels = result.boxes.cls
confidences = result.boxes.conf
for box, label, confidence in zip(boxes, labels, confidences):
  print(f"Bounding Box: {box.tolist()}")
  print(f"Class Label: {label.item()}")
  print(f"Confidence: {confidence.item()}")
```

Milestone 3: training

Now it's time to train our Yolo model:

Activity 1: Building the Model

Training yolo v5 model on a custom dataset.

```
pip install ultralytics
```

```
Collecting ultralytics
  Downloading ultralytics-8.3.40-py3-none-any.whl.metadata (35 kB)
Requirement already satisfied: numpy>=1.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.26.4)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (3.8.0)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.10.0.84)
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (11.0.0)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (6.0.2)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.32.3)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.13.1)
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.5.1+cu121)
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.20.1+cu121)
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.66.6)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.2.2)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.13.2)
Collecting ultralytics-thop>=2.0.0 (from ultralytics)
  Downloading ultralytics_thop-2.0.12-py3-none-any.whl.metadata (9.4 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (4.55.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics) (2024.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2024.8.30)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.16.1)
```

```
from ultralytics import YOLO

model = YOLO('yolov5s.pt')
results = model.train(data="/content/drive/MyDrive/helmet/Hard-Hat-Sample-1/data.yaml",epochs=100)
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      8/100      4.42G      1.238      1.021       1.14         20        640: 100%|██████████| 5/5 [00:01<00:00,  3.66it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  1.95it/s]

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      9/100      4.26G      1.236     0.9284      1.121         41        640: 100%|██████████| 5/5 [00:01<00:00,  2.51it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:01<00:00,  1.13s/it]

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     10/100       4.4G      1.215     0.9328      1.135         63        640: 100%|██████████| 5/5 [00:01<00:00,  3.12it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  3.55it/s]

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     11/100      4.28G      1.171     0.8808      1.125         23        640: 100%|██████████| 5/5 [00:01<00:00,  3.89it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  3.57it/s]

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     12/100      4.22G      1.173     0.8681      1.099         27        640: 100%|██████████| 5/5 [00:01<00:00,  3.83it/s]
```

```
100 epochs completed in 0.087 hours.
Optimizer stripped from runs/detect/train36/weights/last.pt, 18.5MB
Optimizer stripped from runs/detect/train36/weights/best.pt, 18.5MB

Validating runs/detect/train36/weights/best.pt...
Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv5s summary (fused): 193 layers, 9,112,697 parameters, 0 gradients, 23.8 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  4.42it/s]
                   all         20         65       0.97      0.619      0.646      0.473
                  head          3         18      0.941      0.944      0.978      0.703
                helmet         17         45      0.969      0.911       0.96      0.715
                person          1          2          1          0          0          0
Speed: 0.2ms preprocess, 4.3ms inference, 0.0ms loss, 1.2ms postprocess per image
Results saved to runs/detect/train36
```

```
[8]  results = model.val(data="/content/drive/MyDrive/helmet/Hard-Hat-Sample-1/data.yaml",epochs=100);

Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv5s summary (fused): 193 layers, 9,112,697 parameters, 0 gradients, 23.8 GFLOPs
val: Scanning /content/drive/MyDrive/helmet/Hard-Hat-Sample-1/valid/labels.cache... 20 images, 0 backgrounds, 0 corrupt: 100%|
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:01<00:00,  1.60it/s]
                   all         20         65      0.985        0.6      0.647      0.472
                  head          3         18      0.981      0.889      0.979        0.7
                helmet         17         45      0.974      0.911      0.963      0.716
                person          1          2          1          0          0          0
Speed: 0.2ms preprocess, 32.0ms inference, 0.0ms loss, 1.4ms postprocess per image
Results saved to runs/detect/train362
```

Activity 2: Validation

Validating our model using the 'val' folder. Also, we have saved our best.pt Download the model weights from run/detect/train/weights/best.pt

```
results = model.val(data="/content/drive/MyDrive/helmet/Hard-Hat-Sample-1/data.yaml",epochs=100);

Ultralytics 8.3.40 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv5s summary (fused): 193 layers, 9,112,697 parameters, 0 gradients, 23.8 GFLOPs
val: Scanning /content/drive/MyDrive/helmet/Hard-Hat-Sample-1/valid/labels.cache... 20 images, 0 backgrounds, 0 corrupt: 100%|██████████| 20/20 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:01<00:00,  1.60it/s]
                   all         20         65      0.985        0.6      0.647      0.472
                  head          3         18      0.981      0.889      0.979        0.7
                helmet         17         45      0.974      0.911      0.963      0.716
                person          1          2          1          0          0          0
Speed: 0.2ms preprocess, 32.0ms inference, 0.0ms loss, 1.4ms postprocess per image
Results saved to runs/detect/train362
```

Displaying detected image in training notebook:

```
image_path = "/content/drive/MyDrive/helmet/archive/images/BikesHelmets0.png"
results = model(image_path)
result = results[0]
boxes = result.boxes.xyxy
labels = result.boxes.cls
confidences = result.boxes.conf
for box, label, confidence in zip(boxes, labels, confidences):
  print(f"Bounding Box: {box.tolist()}")
  print(f"Class Label: {label.item()}")
  print(f"Confidence: {confidence.item()}")
  results[0].plot()
```

displaying a detected image from the saved folder runs/detect/predict4

```
plt.imshow(results[0].plot())
```

```
<matplotlib.image.AxesImage at 0x7ac90ec73ca0>
```

Milestone 4: Application Building

Now that we have trained our model, let us build our flask application which will be running in our local browser with a user interface. In the flask application, the input parameters are taken

from the HTML page These factors are then given to the model to predict the type of Garbage and showcased on the HTML page to notify the user. Whenever the user interacts with the UI and selects the “Image” button, the next page is opened where the user chooses the image and predicts the output.

Activity 1: Create HTML Pages

- We use HTML to create the front-end part of the web page.
- Here, we have created 3 HTML pages- home.html, intro.html, and upload.html
- home.html displays the home page.
- Intro.html displays an introduction about the project
- upload.html gives the emergency alert For more information regarding HTML https://www.w3schools.com/html/
- We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.
- Link:CSS, JS

Create app.py (Python Flask) file:

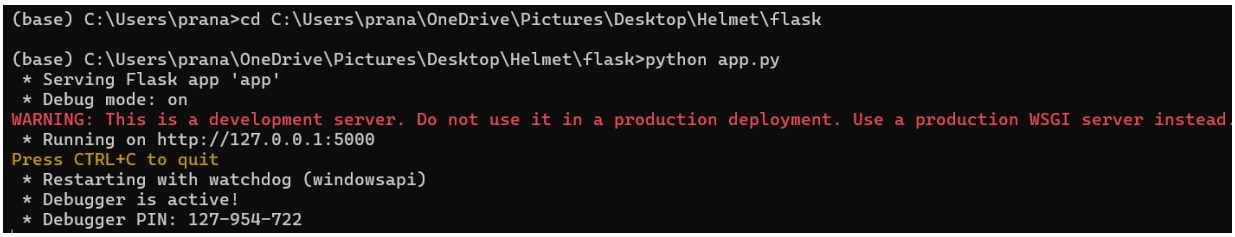
Write the below code in Flask app.py python file script to run the Object Detection Project. To upload image in UI, To display the image in UI:

```
from flask import Flask, Response, render_template,request,redirect, url_for, send_from_directory

import os

import cv2

import pandas as pd

from ultralytics import YOLO # type: ignore

import cvzone

import numpy as np

app = Flask(_name_, template_folder='template')

model = YOLO('best.pt')

def generate_frames():
    count = 0
    while True:
        ret, frame = map.read()
        if not ret:
            break
        count += 1
```

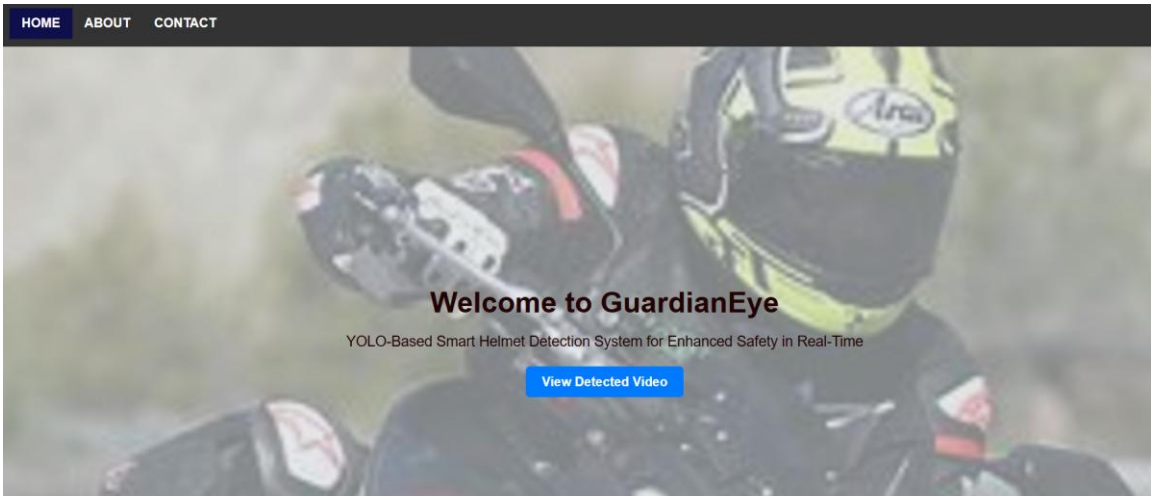
```
        if count % 3 != 0:
            continue
        frame = cv2.resize(frame, (1020,500))
        results = model.predict(frame)
        a = results[0].boxes.data
        px = pd.DataFrame(a).astype("float")
        for index,row in px.iterrows():
            x1 = int(row[0])
            y1 = int(row[1])
            x2 = int(row[2])
            y2 = int(row[3])
            d = int(row[5])
            c = class_list[d] # type: ignore
            cv2.rectangle(frame, (x1,y1), (x2,y2),(255,0,255),2)
            cvzone.putTextRect(frame,f'{c}',(x1,y1),1,1)
            _, buffer = cv2.imencode('.jpg',frame)
            frame = buffer.tobytes()
            yield(b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n'+ frame + b'\r\n')
@app.route('/submit',methods=["POST","GET"])
def submit():
    text = request.form['userInput']
    data = []
    text = re.sub('[^a-zA-Z]',",text.lower()) # type: ignore
    text = text.split()
    text = [ps.stem(w) for w in text if w not in set(stopwords.words('english'))] # type: ignore
    text = ".join(text)
    data.append(text)
    x = vectorizer.transform(data).toarray() # type: ignore
    pred = model.predict(x)
    categories = {0: 'World', 1: 'Sports', 2: 'Business', 3: 'Sci/Tech'}
    output = categories[pred[0]]
    return render_template("output.html", results=f"The predicted news topic is: {output}")
```

```
@app.route('/')
@app.route('/home', methods=['GET','POST'])
def home():
    return render_template('home.html')
@app.route('/output')
def output():
    return render_template('output.html')
@app.route('/contact')
def contact():
    return render_template('contact.html')
@app.route('/about')
def about():
    return render_template('about.html')
@app.route('/video_feed')
def video_feed():
    return Response(generate_frames(),mimetype='multipart/x-mixed-replace; boundary=frame') # type: ignore
if _name_ == '_main_':
    app.run(debug=True)
```

Getting local host in the terminal while running app.py:



Index.html is displayed below:



About Section is displayed below:

GuardianEye

- Home
- About
- Contact

About us

Helmet detection using YOLOv8 likely involves the application of the YOLO version 8 (YOLOv8) object detection algorithm to identify and locate helmets in video frames. YOLOv8 is an evolution of the YOLO algorithm series, known for its real-time object detection capabilities.

Key Features

- **YOLOv8 Architecture:**

  The use of YOLOv11 implies a state-of-the-art object detection algorithm, known for its accuracy and efficiency.

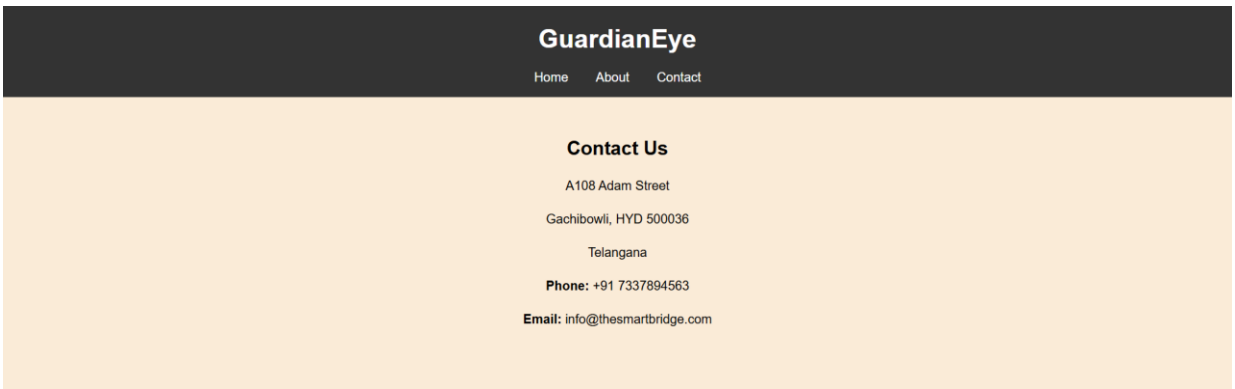
- **Real-Time Detection:**

  YOLOv11 is designed for real-time object detection, making it suitable for applications where quick and accurate identification is crucial.

- **Helmet Detection Model:**

  The system is trained specifically to recognize and detect helmets. This involves training the model on a dataset that includes various images of helmets from different angles, lighting conditions, and environments.

Contact Page is displayed below:



Final Output (after you click on view detected video button) is displayed as follows:

Output: