PROJECT MANUAL

ConstructGuard_ YOLO-Based Safety Gear Surveillance

INTRODUCTION:

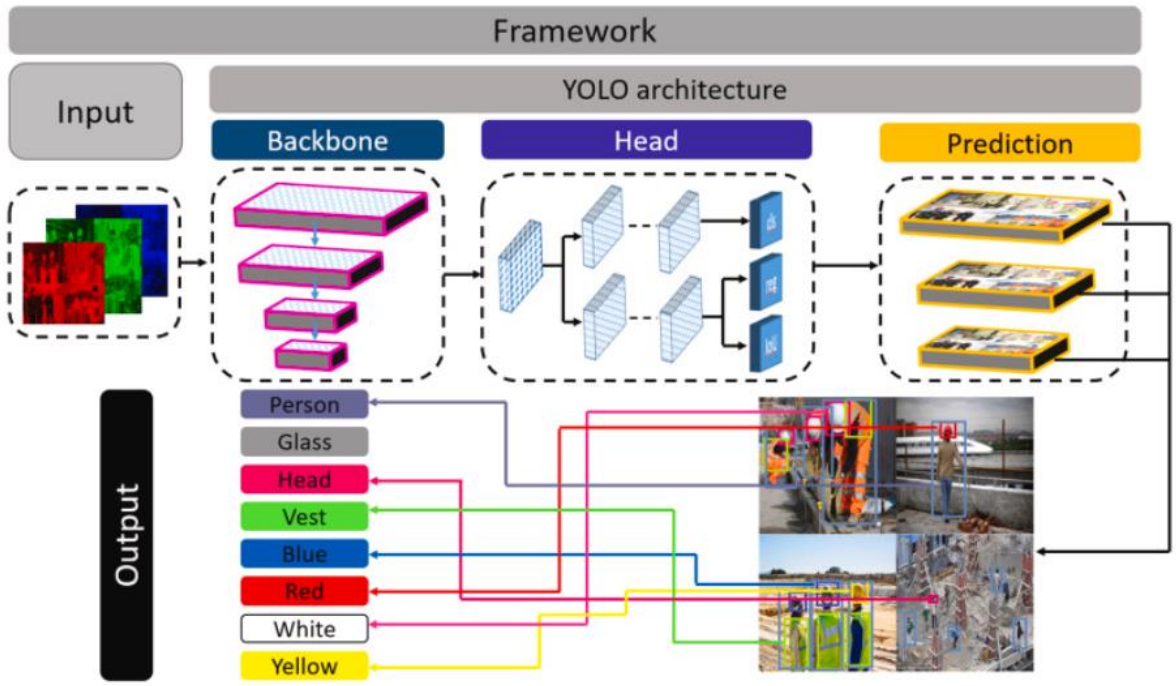
"ConstructGuard: YOLO-Based Safety Gear Surveillance" stands as an innovative application, seamlessly merging computer vision and artificial intelligence to elevate safety and security within construction sites. The backbone of this cutting-edge system lies in the implementation of YOLO (You Only Look Once), a state-of-the-art object detection algorithm. This robust technology excels in precisely identifying and verifying the presence of crucial safety gear donned by construction workers.

With YOLOv8, the latest iteration in the YOLO model series, this system achieves unparalleled efficiency in real-time object detection and classification within the realm of computer vision. YOLOv8's distinctive single-stage detection approach facilitates rapid and accurate identification of safety gear, including hard hats, reflective vests, safety goggles, gloves, and more. Notably, YOLOv8 processes entire images in a single pass, ensuring swift and effective detection even in challenging environmental conditions.

Operational simplicity is a hallmark of the system, with YOLOv8 streamlining the image processing pipeline. The input image is resized to 448×448, subsequently undergoing a single-pass convolutional network operation. The resulting detections are then thresholded based on the model's confidence, embodying a straightforward and efficient approach to safety gear surveillance.

In essence, "ConstructGuard: YOLO-Based Safety Gear Surveillance" redefines safety protocols on construction sites, offering a comprehensive solution that combines the prowess of YOLOv8 with the intricate demands of real-time safety gear detection and verification.

TECHNICAL ARCHITECTURE:



PRE-REQUISTIES:

To complete this project, you must require the following software's, concepts, and Packages. Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It

can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, Spyder, Visual Studio Code. For this project, we will be

using Jupyter notebook and Spyder. To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video **Link:** Click here to watch the video

1. To build Machine learning models you must require the following packages

➢ **Ultralytics:** Ultralytics is a company that creates artificial intelligence models. They

offer cutting-edge solutions for a wide range of AI tasks, including detection, segmentation, classification, tracking, and pose estimation

➢ **Flask:** Web framework used for building Web applications

➢ **Python packages:**

✓ open cmd prompt as administrator

✓ Type pip install ultralytics and click enter.

✓ Type "pip install opencv and click enter.

✓ Type "pip install scikit-learn" and click enter.

✓ Type "pip install flask" and click enter.

DEEP LEARNING CONCEPTS:

1.Object Detection : Object detection is a computer vision technique that identifies and classifies a particular object in a particular setting1. The main goal of object detection is to scan digital images or real-life scenarios to locate instances of every object, separate them, and analyze their necessary features for real-time predictions1.

2. YoloV8: YOLOv8 is the latest version of the YOLO algorithm, developed by Ultralytics1. It is a state-of-the-art model that can be used for object detection, image classification, and instance segmentation

tasks.https://www.youtube.com/watch?v=ag3DLKsl2vk

3. Flask: Flask is a popular Python web framework, meaning it is a third-party.

Python library used for developing web applications.

FLASK BASICS:

If you are using Pycharm IDE, you can install the packages through the command

prompt and follow the same syntax as above.

Project Objectives:

By the end of this project, you will:

4. Know fundamental concepts and techniques of Convolutional Neural Network.

5. Gain a broad understanding of image data.

6. Know how to pre-process/clean the data using different data pre-processing

techniques.

7. know how to build a web application using the Flask framework.

Project Flow:

● The user interacts with the UI (User Interface) to choose the image.

● The chosen image analyzed by the model which is integrated with

flask application.

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

➢ **Data Collection.**

Create Train and Test Folders.

Create data.yaml file

➢ **Training and testing the model**

Save the Model

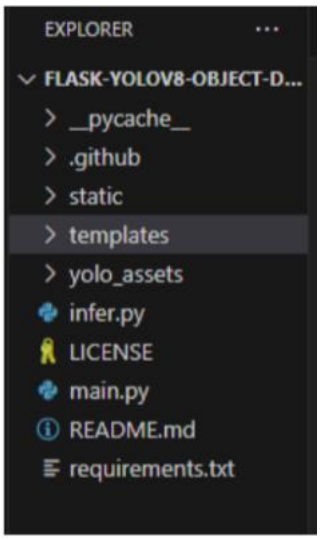
Application Building

Create an HTML file

Build Python Code

PROJECT STRUCTURE:

Create a Project folder which contains files as shown below



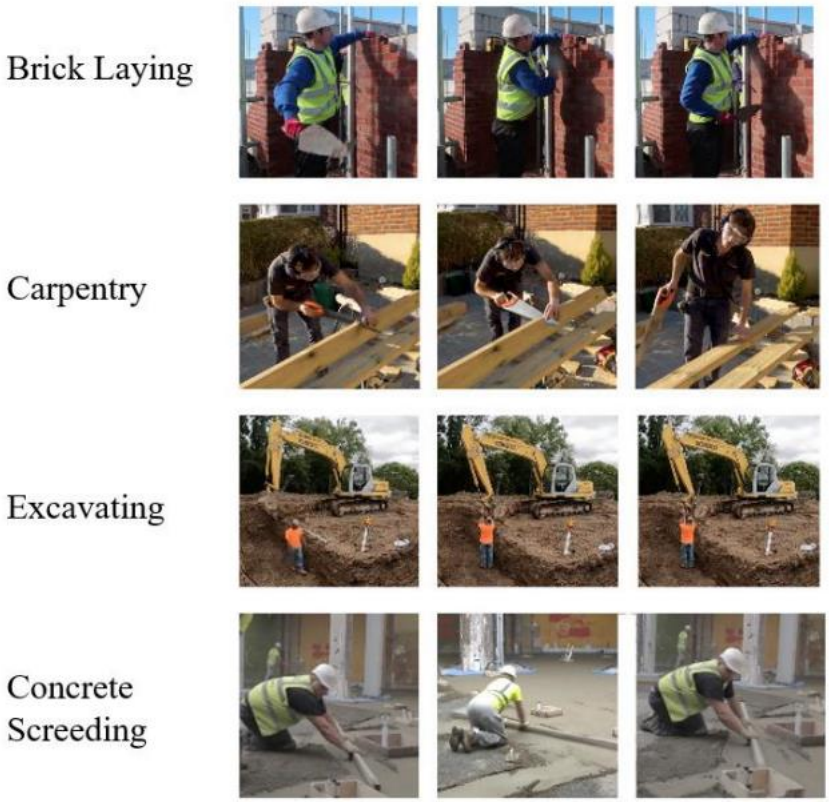
1. The Dataset folder contains the training and testing images for training our model.
2. We are building a Flask Application that needs HTML pages stored in the templates folder and a python script app.py for server-side scripting
3. We need the model which is saved and the saved model in this content is a seanaimal.h5 templates folder containing base.html, index.html pages.

Milestone 1: Collection of Data

Dataset has 3 classes of plastic :
The given dataset has 3 different types of plastics they are following:

- Plastic Bottles
- Plastic Food packaging
- Plastic Bag

Download the dataset from
https://itcon.org/papers/2022_12-ITcon-Bhokare.pdf



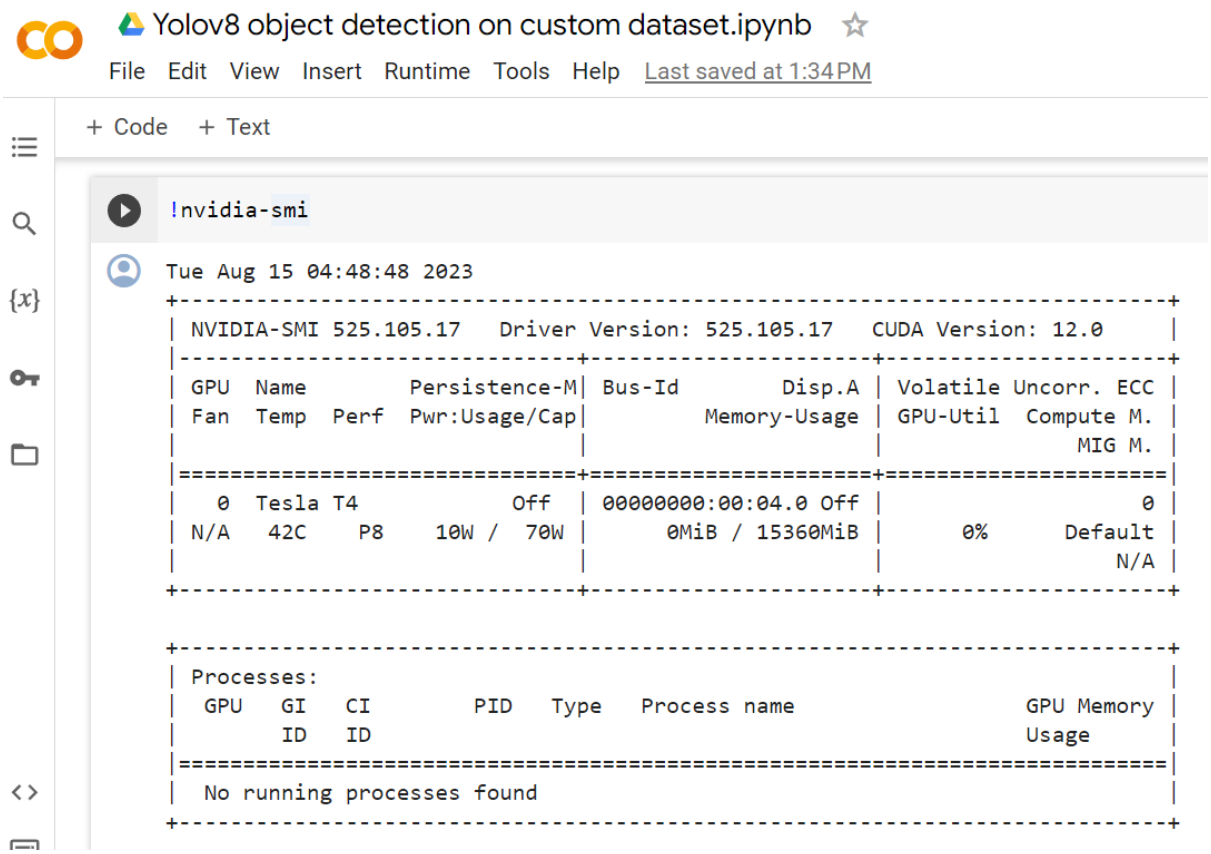
Milestone 2: Image Pre-processing

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.

Milestone 3: training

Now it's time to train our yolo model :

IMPLEMENTATION :



```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[ ] import os
    HOME = os.getcwd()
    print(HOME)
```

```
/content
```

```
# Pip install method (recommended)

!pip install ultralytics==8.0.20

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()
```

```
Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 26.3/78.2 GB disk)
```

```
[ ] from ultralytics import YOLO

    from IPython.display import display, Image
```

```
%cd {HOME}
!yolo task=detect mode=predict model=yolov8n.pt conf=0.25 source='https://media.roboflow.com/notebooks/examples/dog.jpeg'
```

Custom Training

```
[ ] %cd /content/drive/MyDrive/Object detection project
```

```
/content/drive/MyDrive/Object detection project
```

```
%cd /content/drive/MyDrive/Object detection project
%ls
```

```
/content/drive/MyDrive/Object detection project
data/  YoloV8.ipynb
```

```
[ ] !yolo task=detect mode=train model=yolov8s.pt data=data/data.yaml epochs=50 imgsz=224 plots=True
```

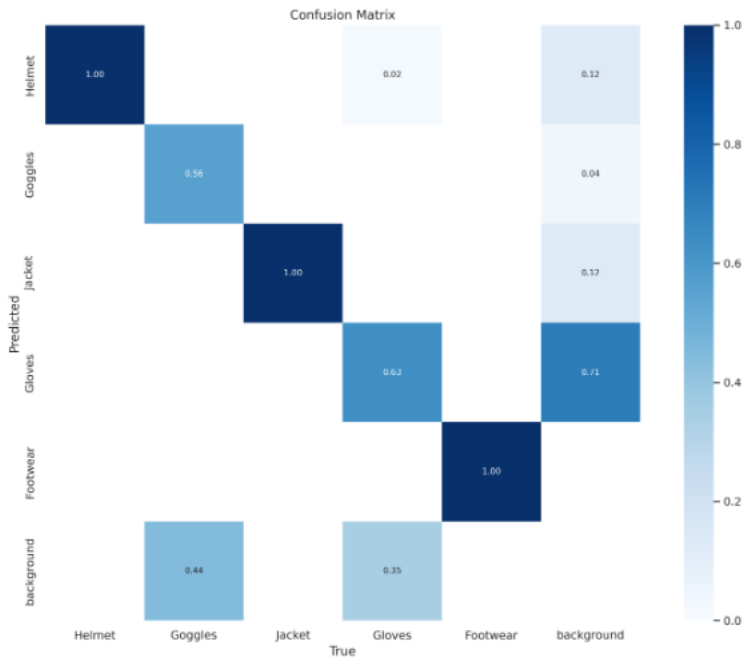
```
Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8s.pt to yolov8s.pt...
100% 21.5M/21.5M [00:00<00:00, 53.8MB/s]

Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
yolo/engine/trainer: task=detect, mode=train, model=yolov8s.yaml, data=data/data.yaml, epochs=50, patience=50, batch=16, imgsz=224, sa
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
100% 755k/755k [00:00<00:00, 2.98MB/s]
2023-08-15 04:52:09.936473: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Overriding model.yaml nc=80 with nc=5

                   from  n    params  module                                       arguments
  0                  -1  1       928  ultralytics.nn.modules.Conv                  [3, 32, 3, 2]
  1                  -1  1     18560  ultralytics.nn.modules.Conv                  [32, 64, 3, 2]
  2                  -1  1     29056  ultralytics.nn.modules.C2f                   [64, 64, 1, True]
  3                  -1  1     73984  ultralytics.nn.modules.Conv                  [64, 128, 3, 2]
  4                  -1  2    197632  ultralytics.nn.modules.C2f                   [128, 128, 2, True]
  5                  -1  1    295424  ultralytics.nn.modules.Conv                  [128, 256, 3, 2]
  6                  -1  2    788480  ultralytics.nn.modules.C2f                   [256, 256, 2, True]
  7                  -1  1   1180672  ultralytics.nn.modules.Conv                  [256, 512, 3, 2]
  8                  -1  1   1838080  ultralytics.nn.modules.C2f                   [512, 512, 1, True]
  9                  -1  1    656896  ultralytics.nn.modules.SPPF                  [512, 512, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11            [-1, 6]  1         0  ultralytics.nn.modules.Concat                [1]
 12                  -1  1    591360  ultralytics.nn.modules.C2f                   [768, 256, 1]
```

```
50 epochs completed in 0.067 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 22.5MB
Optimizer stripped from runs/detect/train/weights/best.pt, 22.5MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11127519 parameters, 0 gradients, 28.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100
                   all         31        101      0.766       0.82      0.855      0.537
                Helmet         31         18      0.773          1      0.992      0.699
               Goggles         31          9      0.814      0.556       0.65      0.301
                Jacket         31         14      0.659      0.929      0.972      0.734
                Gloves         31         52      0.666      0.615      0.667      0.407
              Footwear         31          8      0.917          1      0.995      0.547
Speed: 0.0ms pre-process, 1.1ms inference, 0.0ms loss, 1.2ms post-process per image
Results saved to runs/detect/train
```

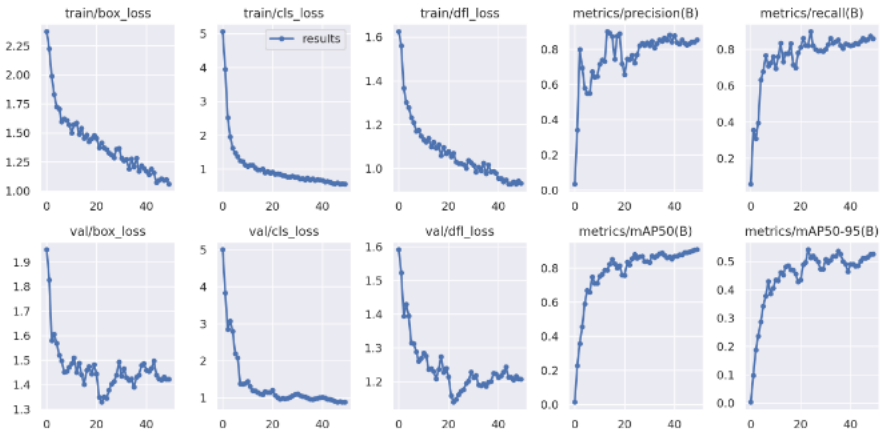
```
[ ] !ls runs/detect/train/
```

```
args.yaml                                        train_batch0.jpg
confusion_matrix.png                             train_batch1.jpg
events.out.tfevents.1692075131.1fa0f6a84ea8.1738.0  train_batch2.jpg
F1_curve.png                                     train_batch320.jpg
P_curve.png                                      train_batch321.jpg
PR_curve.png                                     train_batch322.jpg
R_curve.png                                      val_batch0_labels.jpg
results.csv                                      val_batch0_pred.jpg
results.png                                      weights
```

```
Image(filename='runs/detect/train/confusion_matrix.png', width=600)
```



```
[ ]  Image(filename='runs/detect/train/results.png', width=600)
```




```
Image(filename='runs/detect/train/val_batch0_pred.jpg', width=600)
```

Testing Custom Model

```
!pip install ultralytics==8.0.20
```

```
Collecting ultralytics==8.0.20
  Downloading ultralytics-8.0.20-py3-none-any.whl (261 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 261.2/261.2 kB 4.9 MB/s eta 0:00:00
Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (3.7.1)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (1.23.5)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (4.8.0.76)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (9.4.0)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (6.0.1)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (2.31.0)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (1.10.1)
Requirement already satisfied: torch>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (2.0.1+cu118)
Requirement already satisfied: torchvision>=0.8.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (0.15.2+cu118)
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (4.66.1)
Requirement already satisfied: tensorboard>=2.4.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (2.12.3)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (1.5.3)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (0.12.2)
Requirement already satisfied: ipython in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (7.34.0)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from ultralytics==8.0.20) (5.9.5)
Collecting thop>=0.1.1 (from ultralytics==8.0.20)
  Downloading thop-0.1.1.post2209072238-py3-none-any.whl (15 kB)
Collecting sentry-sdk (from ultralytics==8.0.20)
  Downloading sentry_sdk-1.29.2-py2.py3-none-any.whl (215 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 215.6/215.6 kB 10.2 MB/s eta 0:00:00
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics==8.0.20) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics==8.0.20) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics==8.0.20) (4.42.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics==8.0.20) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics==8.0.20) (23.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics==8.0.20) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics==8.0.20) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics==8.0.20) (2023.3)
```

```python
from google.colab import drive
drive.mount('/content/drive')

from ultralytics import YOLO
import cv2
import torch
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import warnings

warnings.filterwarnings('ignore')

class_names = ['Helmet', 'Goggles', 'Jacket', 'Gloves', 'Footwear']

def custom_detect(img_path:str, score:int=0.4):
    model = YOLO('/content/drive/MyDrive/Object detection project/runs/detect/train/weights/b
    result = model.predict(source=img_path, conf=score)
    img = cv2.imread(img_path)

    clsIdx = torch.tensor(result[0].boxes.cls, dtype=torch.int32).tolist()
    bboxs = torch.tensor(result[0].boxes.xyxy, dtype=torch.int32).tolist()
    confs = torch.tensor(result[0].boxes.conf, dtype=torch.float16).tolist()

    for idx, box, conf in zip(clsIdx, bboxs, confs):
        classname = class_names[idx]
        print('class name : ', classname, conf)

        bbox = (box[0], box[1], box[2]-box[0], box[3]-box[1])
        cv2.rectangle(img, bbox, color=(255, 0, 0), thickness=2)
        cv2.putText(img, '{0}_{1:.2f}'.format(classname, conf), \
                        (box[0]+5, box[1]+20), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 1)

    img = cv2.resize(img, (700, 600))
```

```
    plt.show()

    return clsIdx, confs

custom_detect('/content/drive/MyDrive/Object detection project/data/test/images/1.jpeg')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11127519 parameters, 0 gradients, 28.4 GFLOPs
class name :  Gloves 0.640625
class name :  Helmet 0.56982421875
class name :  Gloves 0.54541015625
class name :  Gloves 0.427978515625
```


```
([3, 0, 3, 3], [0.640625, 0.56982421875, 0.54541015625, 0.427978515625])
```

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 know 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

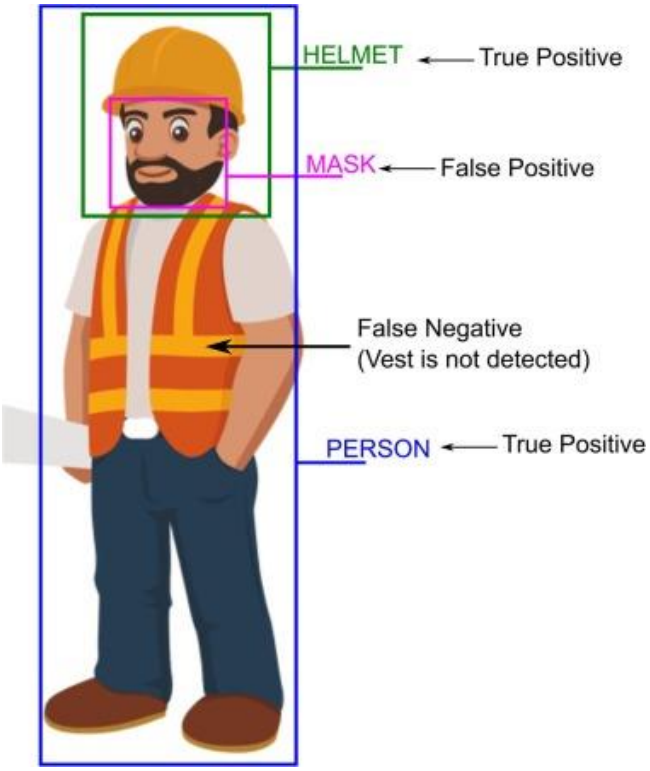
Create app.py (Python Flask) file :-

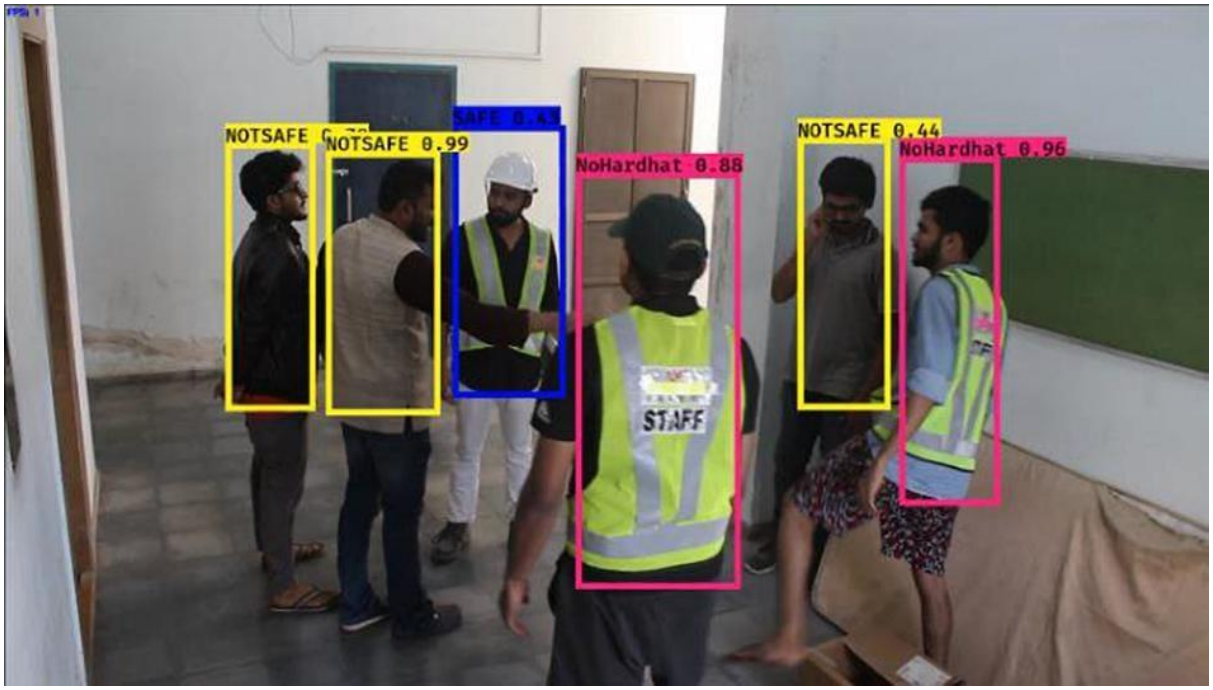
main.py

```
1  from flask import Flask, render_template, Response, session
2
3  from flask wtf import FlaskForm Import secrets
4
5  from wtforms import FileField, SubmitField
6
7  from wtforms.validators import Input Required
8
9  from werkzeug.utils import secure_filename
10
11 from infer import *
12
13 app Flask (name_)
14 app.secret_key secrets.token_hex(16)
15
16 output_directory 'yolo_assets/Uploads
17 if not os.path.exists(output_directory):
18    os.makedirs(output_directory),
19 app.config['UPLOAD_FOLDER'] output_directory
20
21 class UploadFileForm(FlaskForm):
22
23       Represents the form to uploaa video file for object detection.
24
25       file FileField("File", validators=[InputRequired()]) submit Subm
```

Infer.py

```
import cv2
import math
import numpy as np
from ultralytics import YOLO
from seaborn import color_palette
import os

def load_class_names(file_name):

    Returns a list of class names read from the file 'file_name'.

    Args:

        fi1ename(str) : The path to the file containing the class na

    Returns:

        List[str]: A list of class names.

    with open(file_name, 'r') as f:
        class_names f.read().splitlines() return class_names

def draw_bbox(frame, boxes, class_names, colors):

    Draws bounding boxes with labels on the input frame.
```




NOTSAFE 0.99
NoHardhat 0.88
NOTSAFE 0.44
NoHardhat 0.96
STAFF


Face, 1.00
Face, 1.00
Face, 1.00


HARD HAT - 1
HARD HAT - 6
SAFETY VEST - 2
SAFETY VEST - 4
GLOVES OFF - 5
GLOVES ON - 3


Hat
Hat
No Hat
No Hat
Hat


Object Detection Feed from Video
Person (0.9%)
Safety Vest (0.89
Choose File
No file chosen
Submit
Return to Home