ASSIGNMENT-4

Name: Hitarth Paliwal

Roll No: 68

Batch: A-4

Analyze, design, and optimize an object detection and multi-object classification model by integrating detection, segmentation, and recognition tasks. Evaluate model performance on real-world data and explore potential applications.

Task 1: Environment Setup and YOLOv11 Installation

Objective: Set up the required libraries and dependencies to run YOLOv11.

Instructions:

1. Install Python and required libraries (PyTorch, OpenCV, Ultralytics, etc.).
2. Install YOLOv11 from the official repository.
3. Verify the installation by running a sample script.
4. **Expected Outcome:** A functional YOLOv11 environment ready for experimentation

```
# Code Task 1
# Install required libraries
!pip install roboflow ultralytics torch torchvision opencv-python

# Import necessary modules
from roboflow import Roboflow
import torch
import ultralytics

# Check if GPU is available
print("PyTorch GPU Available:", torch.cuda.is_available())

# Verify YOLO installation
!yolo --version
```

Task 2: Dataset Preparation & Preprocessing Objective: Load and preprocess a dataset for object detection.

Instructions:

1. Choose a Dataset – Use COCO, Pascal VOC, or a custom dataset.
2. Annotate Images – If using a custom dataset, label objects using Roboflow or LabelImg.
3. Convert Annotations – Use Roboflow to export the dataset in YOLO format.
4. Download the Dataset – Use the Roboflow API to fetch the dataset.
5. Split the Dataset – Divide into train (80%), validation (10%), and test (10%).
6. **Expected Outcome:** A well-structured dataset in YOLO format.

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="zeoNhq7ZawrpQeLMPwcJ")
project =
rf.workspace("plasticpollution").project("plastic_detection-
hgahq")
version = project.version(2)
dataset = version.download("yolov11")
```

Task 3: Training YOLOv11 Model Objective: Train YOLOv11 on the prepared dataset.

Instructions:

1. Configure the training parameters (batch size, epochs, learning rate).
2. Train the YOLOv11 model using the dataset.
3. Monitor training progress (loss, accuracy, mAP).
4. Save the trained model weights.
5. Expected Outcome: A trained YOLOv11 model ready for inference.

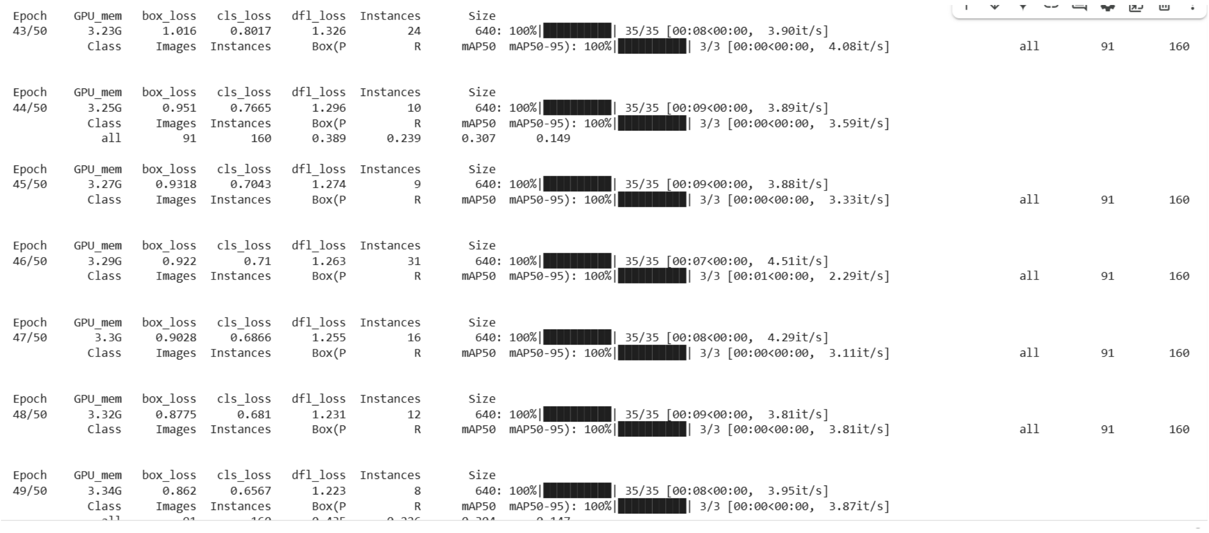
```
#code for task 3
# Verify dataset structure
import os
print("Dataset files:",
os.listdir("/content/plastic_detection-2"))
```

```
!pip install ultralytics

from ultralytics import YOLO

model = YOLO("yolov8n.pt")  # small and fast for testing

model.train(data="/content/plastic_detection-2/data.yaml",
epochs=50)
```



```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
43/50      3.23G      1.016     0.8017      1.326         24        640: 100%|██████████| 35/35 [00:08<00:00,  3.90it/s]
             Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00,  4.08it/s]                   all         91        160

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
44/50      3.25G      0.951     0.7665      1.296         10        640: 100%|██████████| 35/35 [00:09<00:00,  3.89it/s]
             Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00,  3.59it/s]
               all         91        160      0.389      0.239      0.307      0.149

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
45/50      3.27G     0.9318     0.7043      1.274          9        640: 100%|██████████| 35/35 [00:09<00:00,  3.88it/s]
             Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00,  3.33it/s]                   all         91        160

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
46/50      3.29G      0.922       0.71      1.263         31        640: 100%|██████████| 35/35 [00:07<00:00,  4.51it/s]
             Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:01<00:00,  2.29it/s]                   all         91        160

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
47/50       3.3G     0.9028     0.6866      1.255         16        640: 100%|██████████| 35/35 [00:08<00:00,  4.29it/s]
             Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00,  3.11it/s]                   all         91        160

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
48/50      3.32G     0.8775      0.681      1.231         12        640: 100%|██████████| 35/35 [00:09<00:00,  3.81it/s]
             Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00,  3.81it/s]                   all         91        160

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
49/50      3.34G      0.862     0.6567      1.223          8        640: 100%|██████████| 35/35 [00:08<00:00,  3.95it/s]
             Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00,  3.87it/s]
```

Task 4: Model Inference and Evaluation Objective: Test the trained model on new images and videos.

Instructions:

1. Load the trained model weights.
2. Run object detection on test images and videos.
3. Evaluate the model performance using mAP (mean Average Precision), precision, recall.

✅ Mean Average Precision (mAP@50, mAP@50-95) – Measures model accuracy across different IoU thresholds.

✅ Precision & Recall – Evaluates the tradeoff between false positives and false negatives.

✅ F1 Score – Balances precision and recall for a comprehensive model assessment.

Discuss the results in detail

4. Visualize results with bounding boxes.
5. Expected Outcome: Detection results with bounding boxes and performance metrics.

```
# Code for Task 4
# Run inference on test images
!yolo task=detect mode=predict
model=/content/runs/detect/train/weights/best.pt \
      source=/content/plastic_detection-2/train/images
```

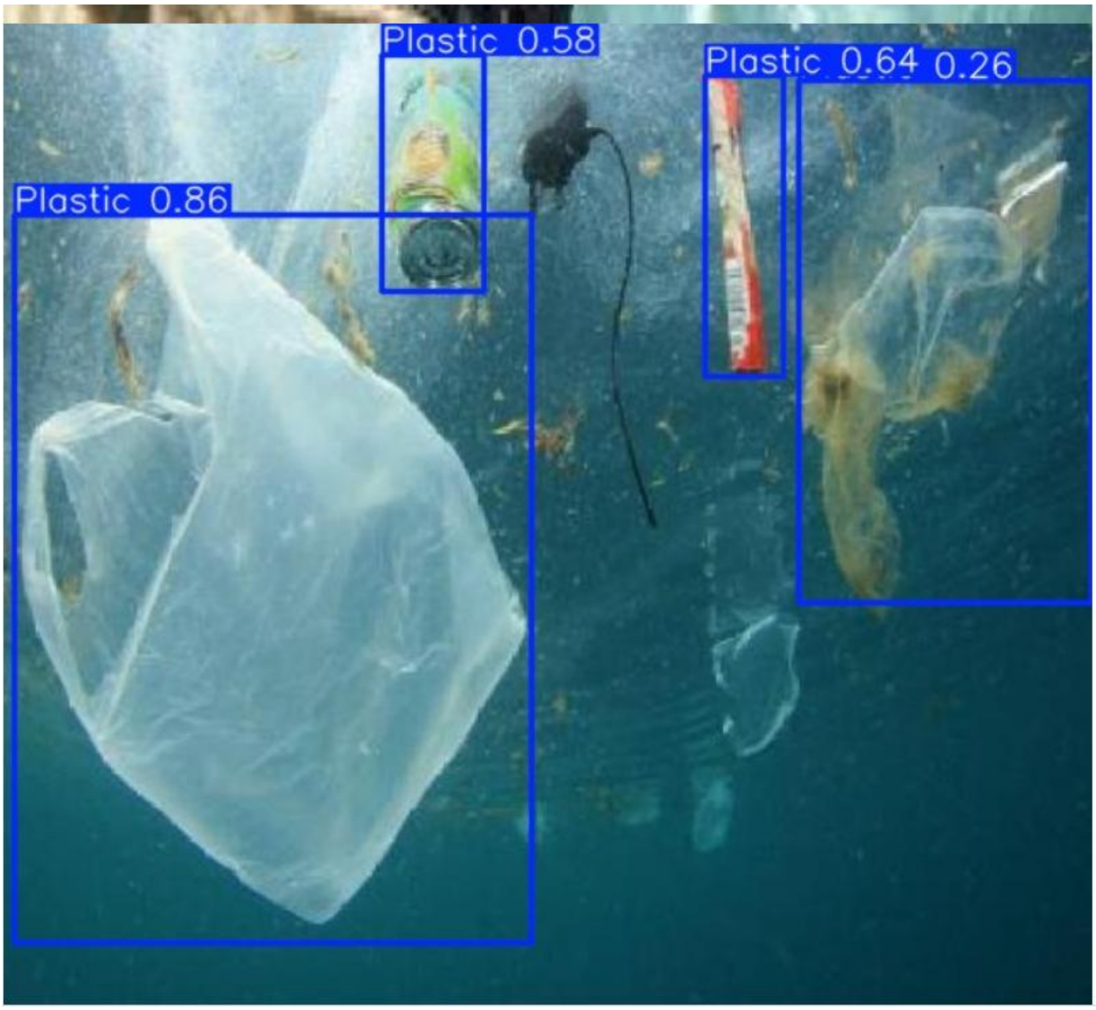
```
# Evaluate Model Performance
!yolo task=detect mode=val
model=/content/runs/detect/train/weights/best.pt \
      data=/content/plastic_detection-2/data.yaml
```

```
!yolo task=detect mode=predict \
      model=/content/yolo11s.pt \
      source=/content/plastic_detection-2/train/images \
      imgsz=640 \
      conf=0.25 \
      save=True
```

```
from IPython.display import display
from PIL import Image
import os

pred_dir = "/content/runs/detect/predict/"

for img in os.listdir(pred_dir):
    if img.endswith((".jpg", ".png")):
        display(Image.open(os.path.join(pred_dir, img)))
```



Declaration:

I, Hitarth Paliwal, confirm that the work submitted in this assignment is my own and has been completed following academic integrity guidelines. The code is uploaded on my GitHub repository account, and the repository link is provided below:

GitHub Repository Link: [Insert GitHub Link]

Signature: [Hitarth Paliwal]