Detection and Classification of sign objects

Code walkthrough:

What the code does

This code performs the following steps:

1. Reads annotation data from a CSV file containing bounding box info.
2. Converts the bounding boxes into YOLO format.
3. Saves YOLO labels into .txt files.
4. Splits the dataset into training and validation sets.
5. Copies images and labels into the required YOLO directory structure.
6. Creates a dataset.yaml config file for YOLO.
7. Trains a YOLOv8 model using the Ultralytics library.

Annotation CSV

▼

Read CSV into DataFrame

▼

Rename Columns for Clarity

▼

Define Class Names and Map to Indices

▼

Loop through each row

- Get image name, bounding box, class
  - Read image using PIL
- Calculate YOLO format (x_center, y_center, width, height)
  - Write to label .txt file

▼

YOLO Labels Folder created (e.g., `./labels`)

▼

Shuffle and Split Images into Train & Validation (80:20)

▼

Create Folder Structure:

- ./dataset/images/train
- ./dataset/images/val
- ./dataset/labels/train
- ./dataset/labels/val

▼

Copy images & corresponding label .txt to respective folders

▼

Create `dataset.yaml` file

- Paths to train/val image folders
- Number of classes (nc)
- Class names

▼

Train YOLOv8 Model using:

- yolov8n.pt
- dataset.yaml
- 50 epochs, image size 416, batch size 4

Tools Used

Tools	Purpose
Pandas	Read and process the CSV file
os, shutil	File handling and folder creation
PIL.Image	Get image dimensions for bounding box conversion
random	Shuffle the dataset
yaml	Create YOLO format config file
ultralytics.YOLO	Train the YOLOv8 model

Exploratory data analysis on the data:

I have used df.head() to display first few rows in the Annotation.csv file and df.isnull().sum() to check for missing values

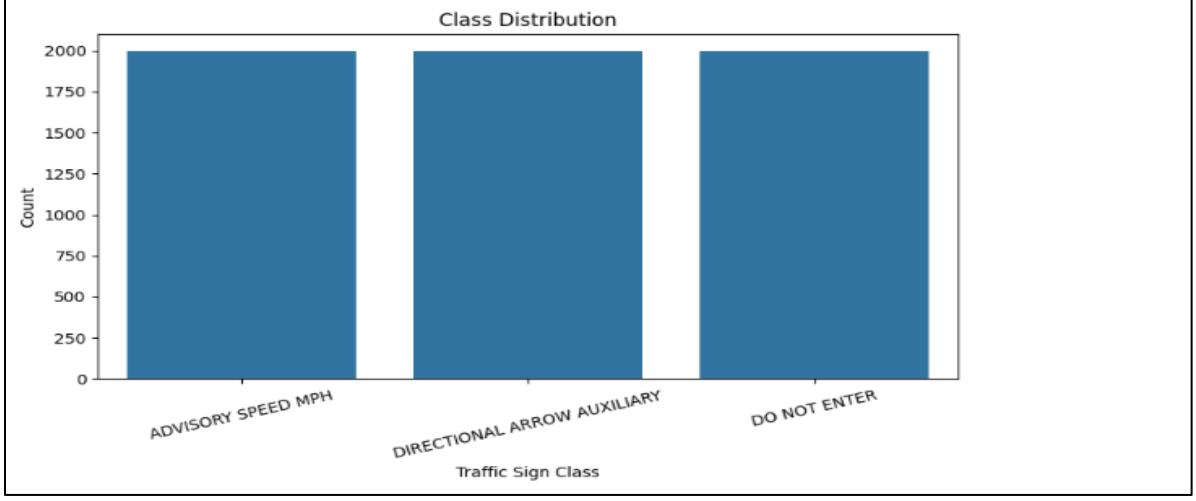
```
import pandas as pd

df = pd.read_csv("C:/Users/user/Downloads/Assignment-20250409T022654Z-001/Assignment/Annotation/Annotation_For3Class.csv")
print(df.head())
print(df.isnull().sum())
```

```
          ImagePath    X0    X1   Y0   Y1              Class
0  000015335564.jpg    80   116  477  514  ADVISORY SPEED MPH
1  000015340509.jpg  1720  1760  485  526  ADVISORY SPEED MPH
2  000016400582.jpg    50    76  468  490  ADVISORY SPEED MPH
3  000016450565.jpg  1765  1840  517  590  ADVISORY SPEED MPH
4  000024250089.jpg   354   428  430  504  ADVISORY SPEED MPH
ImagePath    0
X0           0
X1           0
Y0           0
Y1           0
Class        0
dtype: int64
```

To view if the Annotation.csv file has any class imbalance EDA used

```
# Class distribution
plt.figure(figsize=(8, 5))
sns.countplot(x='class', data=df)
plt.title('Class Distribution')
plt.xlabel('Traffic Sign Class')
plt.ylabel('Count')
plt.xticks(rotation=15)
plt.tight_layout()
plt.show()
```



To check for duplicate annotation and invalid bounding boxes, the below code is used, wherein 18 duplicate annotations where found and deleted in later stages

```python
# Duplicate annotations
duplicates = df.duplicated()
print(f"\n Total Duplicate Annotations: {duplicates.sum()}")
# Invalid bounding boxes (zero or negative size)
invalid_boxes = df[(df['bbox_width'] <= 0) | (df['bbox_height'] <= 0)]
print(f" Invalid Bounding Boxes (Width/Height <= 0): {len(invalid_boxes)}")
```

```
 Total Duplicate Annotations: 18
 Invalid Bounding Boxes (Width/Height <= 0): 0
```

Number of training epochs and other parameters set for training

Configuration set in dataset.yaml file

```yaml
names:
- DIRECTIONAL ARROW AUXILIARY
- DO NOT ENTER
- ADVISORY SPEED MPH
nc: 3
train: C:/Users/user/Downloads/Assignment-20250409T022654Z-001/Assignment/Dataset/images/train
val: C:/Users/user/Downloads/Assignment-20250409T022654Z-001/Assignment/Dataset/images/val

```

Parameters set for training the model

```python
from ultralytics import YOLO

# Load base YOLOv8 model
model = YOLO("yolov8n.pt")

# Train on custom dataset
model.train(
    data="dataset.yaml",
    epochs=50,
    imgsz=416,
    batch=4,
    workers=2,
    seed=42
)
```

Where:

Model = yolov8n.pt;

data = dataset.yaml; contains the Dataset configuration file

epochs = 50; Total number of training cycles through the dataset.

Imgsz=416; input image size (Height x Width)

Batch = 4; total number of images processed in one training step, have used low batch since it uses less RAM; we can even set it to 8 or 16 based on our RAM capacity

Workers = 2; number of CPU threads used for parallel processing

Seed = 42; used for reproducibility

I have used yolov8n.pt for training, which is the smallest and fastest model in the YOLOv8 family, as this model can be trained quickly without the need of a GPU. For more accurate results, larger models like yolov8s.pt, yolov8m.pt, yolov8l.pt, and yolov8x.pt can be used. These models offer higher accuracy but come at the cost of increased model size and longer training time, especially on a CPU. Additionally, for high-end systems with good GPU and RAM configurations, even more powerful models and frameworks can be considered, such as:YOLO-NAS, Detectron2 (Mask R-CNN), Scaled-YOLOv4.These models provide state-of-the-art accuracy and support advanced capabilities like instance segmentation, but they require significantly more hardware resources to train and deploy efficiently.

A note on the Training results regarding the performance and graphs.

The object detection model was trained for 50 epochs using YOLOv8 on a multi-class dataset containing traffic signs. The training spanned approximately 8.5 hours, and the final results reflect strong generalization with high precision and recall across classes.

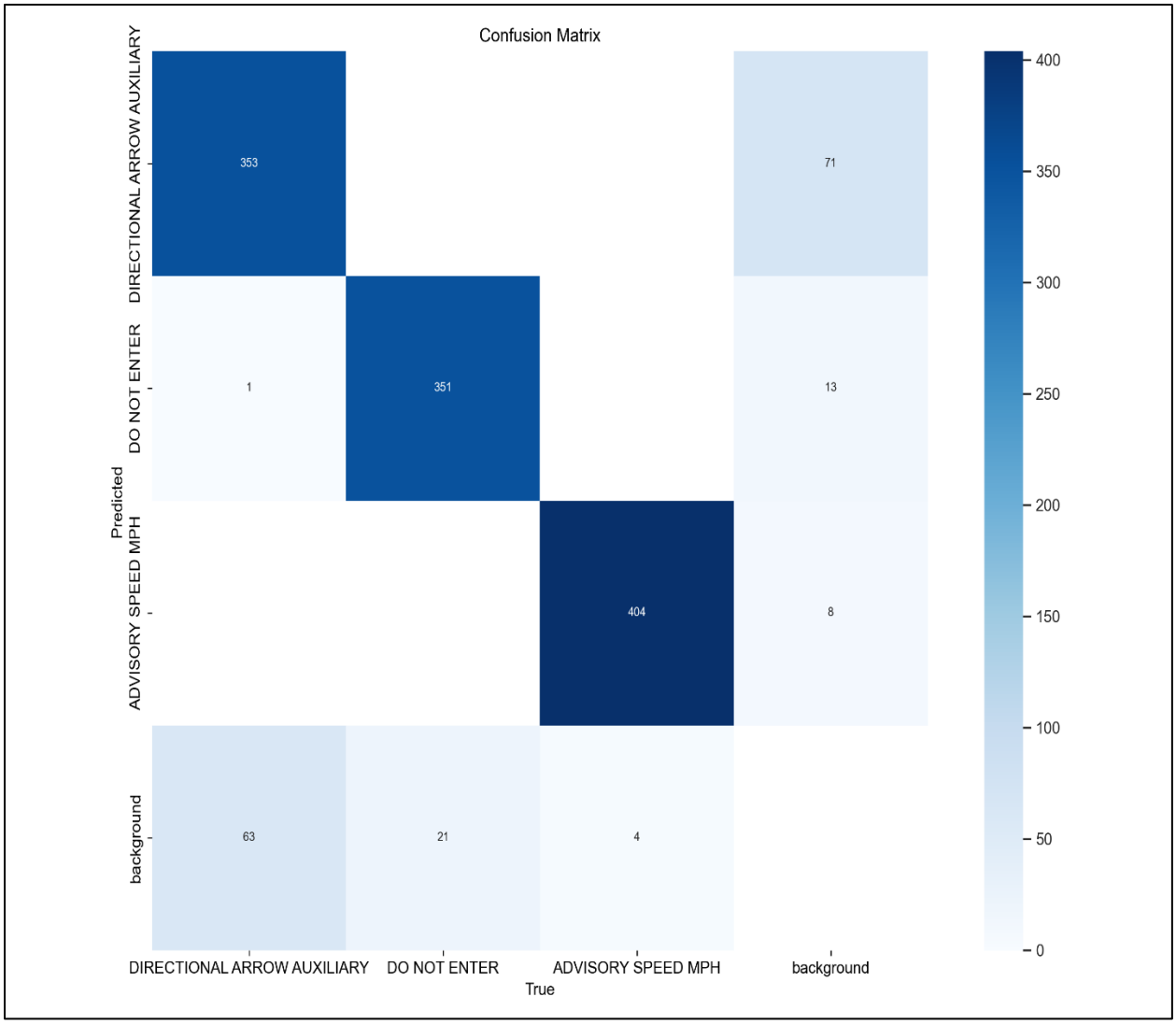
- Overall Metrics:
  - Precision: 0.919
  - Recall: 0.926
  - mAP@0.5: 0.946
  - mAP@0.5:0.95: 0.801
- Per-Class mAP@0.5:
  - Advisory Speed MPH: 0.995
  - Do Not Enter: 0.975
  - Directional Arrow Auxiliary: 0.867

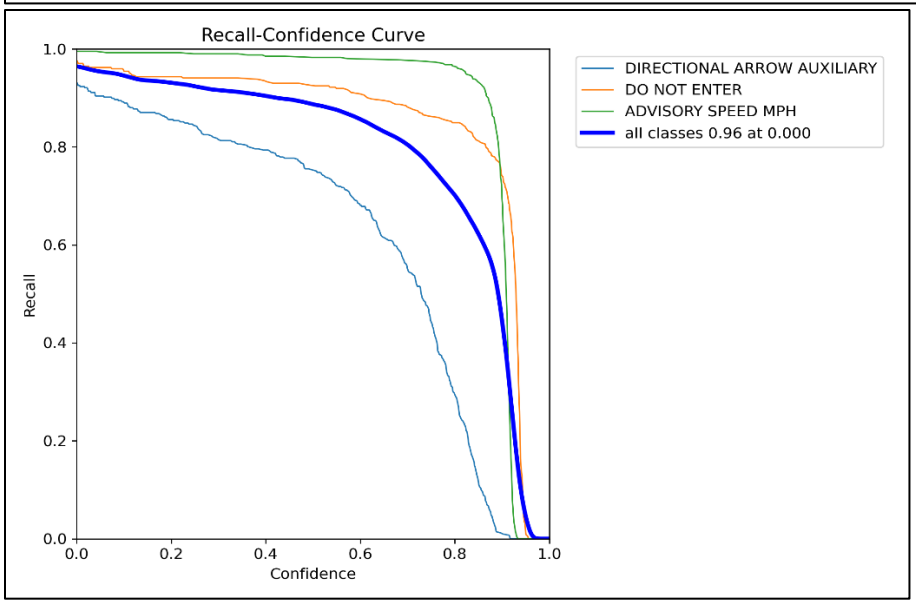
These scores reflect high confidence in predictions for all three classes, especially for "Advisory Speed MPH" signs.

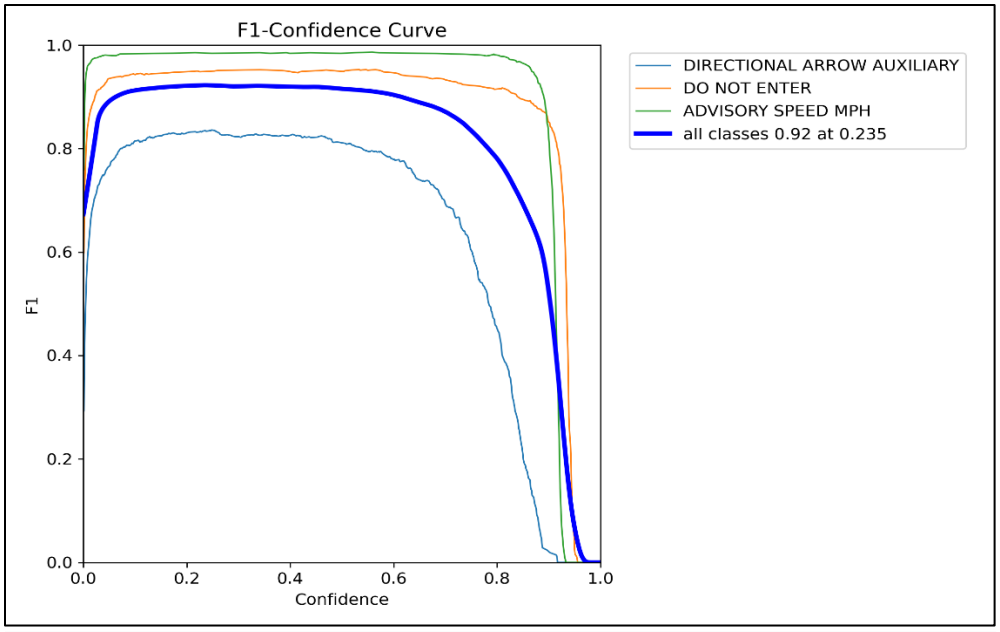
Loss values decreased smoothly, and no overfitting was observed. Validation metrics remained closely aligned with training trends.

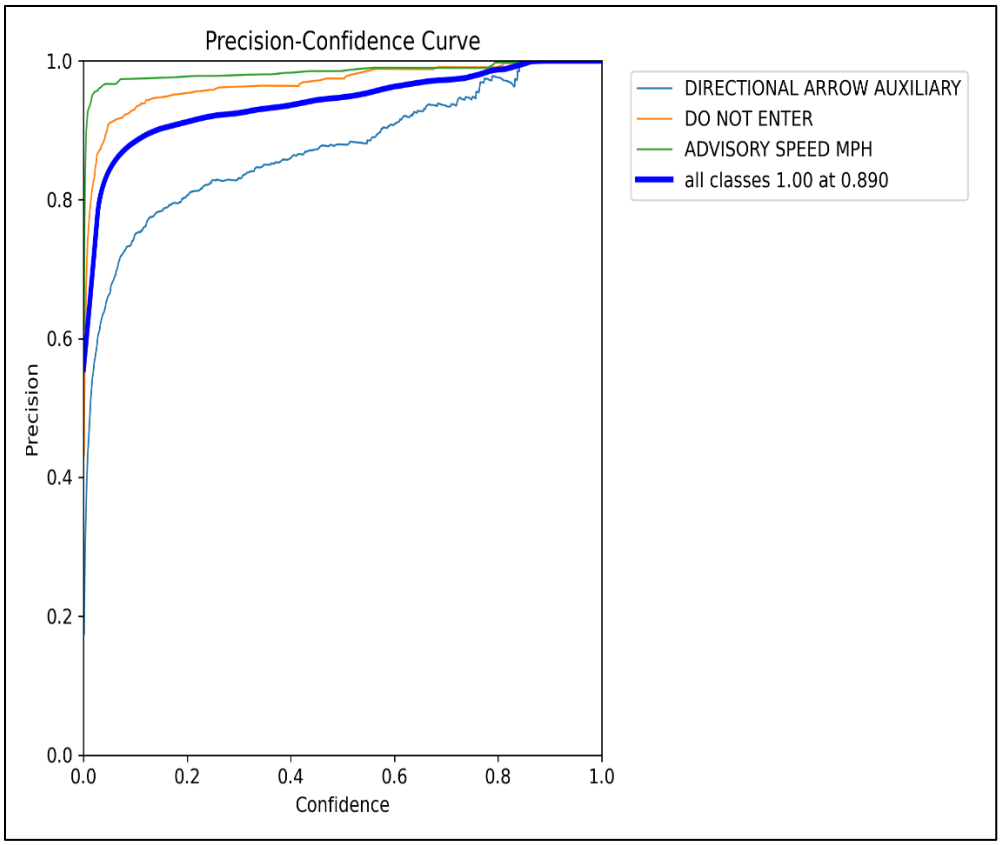
Final model (best.pt) is 6.2MB — very lightweight.

Inference speed is 19.3ms/image on CPU, making it real-time capable for embedded or edge applications.


Confusion Matrix
DIRECTIONAL ARROW AUXILIARY
DO NOT ENTER
ADVISORY SPEED MPH
background
Predicted
353
71
1
351
13
404
8
63
21
4
DIRECTIONAL ARROW AUXILIARY
DO NOT ENTER
ADVISORY SPEED MPH
background
True
400
350
300
250
200
150
100
50
0


Recall-Confidence Curve
DIRECTIONAL ARROW AUXILIARY
DO NOT ENTER
ADVISORY SPEED MPH
all classes 0.96 at 0.000
Recall
Confidence
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


F1-Confidence Curve
DIRECTIONAL ARROW AUXILIARY
DO NOT ENTER
ADVISORY SPEED MPH
all classes 0.92 at 0.235
F1
Confidence
0.0
0.2
0.4
0.6
0.8
1.0


Precision-Confidence Curve
DIRECTIONAL ARROW AUXILIARY
DO NOT ENTER
ADVISORY SPEED MPH
all classes 1.00 at 0.890
Precision
Confidence
0.0
0.2
0.4
0.6
0.8
1.0

A note on the detection results obtained on the unseen data

When the unseen data was passed as an input to the model the model was able to detect a class in 128 frames out of 258 frames and rest 129 times the model was not able to detect any class this could be due to many reasons such as distance of the sign board trained and the one in the video, sign bords having low illumination or shadow formation on the sign board in few frames as it few videos where captured from far distance it was not able to identify the sign board until it came near by and so on this could be overcome by complete it

When the model was evaluated on unseen data, it successfully detected a class in 128 out of 258 frames, while in the remaining 130 frames, it failed to detect any class. This detection gap can be due to several factors:

- Distance of the Sign Board: The model was primarily trained on images captured at closer distances. In some test videos, sign boards appeared far from the camera. As a result, the model struggled to recognize them until they came closer.
- Low Illumination and Shadows: Some frames suffered from poor lighting conditions or shadow formation on the signboards. These variations in lighting, which were less represented in the training data, negatively impacted the model's ability to generalize.
- Angle and Obstruction: In certain instances, the signboards were captured from challenging angles, which reduced the visibility of key features required for correct classification.

Potential Solutions:

- Enhancing the diversity of the training dataset by incorporating images of sign boards under different lighting conditions, distances, and viewing angles.
- Applying data augmentation techniques to simulate such variations.
- Using object detection models with multi-scale feature extraction capabilities to better handle signs of varying sizes and positions.

The output video is attached in Mail

A note on the object tracking results

Out of 258 frames, the object tracker technique has reported 0 in most of the frames. This can be due to many reason

- Model not being able to detect the object continuously
- Objects are entering/leaving the frame quickly
- Objects are not present in most frames.

Solution:

- Building a reliable model using state of the art techniques