Vehicle & Pedestrian Object Tracker – Project Report

1. Introduction

The goal of this project was to develop an object tracker for vehicle, pedestrian, road, and two-wheeler using YOLOv8-Seg for detection and ByteTrack for multi-object tracking. The system can track objects in videos and export results in a structured JSON format with object IDs, classes, bounding boxes, and frame numbers.

2. Tools and Frameworks Used

Python 3.10, Django 5.2, Ultralytics YOLOv8, ByteTrack/BoT-SORT, Labellerr, OpenCV, PIL, NumPy, Matplotlib / Seaborn .

3. Dataset Collection & Annotation

Raw dataset: 150–200 images containing vehicle, pedestrian, road, and two-wheeler.
Annotated dataset: 94 images labeled in Labellerr with polygon masks for four classes.
Test dataset: 20 images for prediction and 20 images for evaluation.
Annotation export: YOLO-compatible format for training.

4. Model Training

Model: YOLOv8-Seg (nano). Epochs: 100. Batch size: 16. Image size: 640x640. Optimizer: auto. AMP: Enabled. Training split: 94 images for training/validation, 20 images for prediction/testing. Data augmentations: Mosaic, horizontal flip, HSV adjustments. Logging metrics: Precision, Recall, F1-score, mAP.

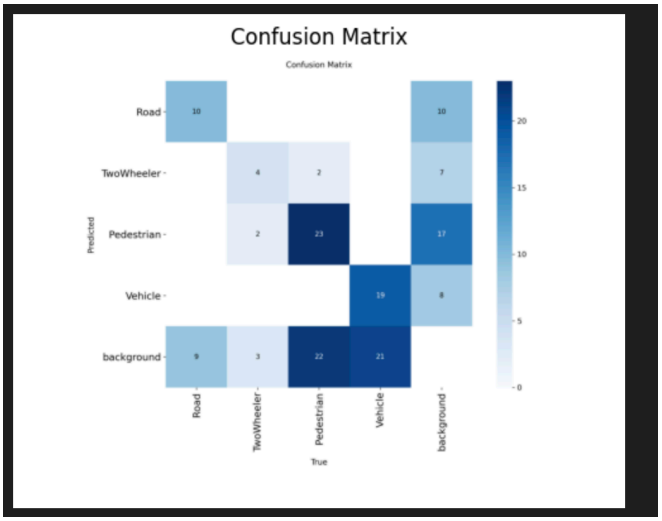
5. Model Evaluation

Evaluation dataset: 20 images not used in training/validation.

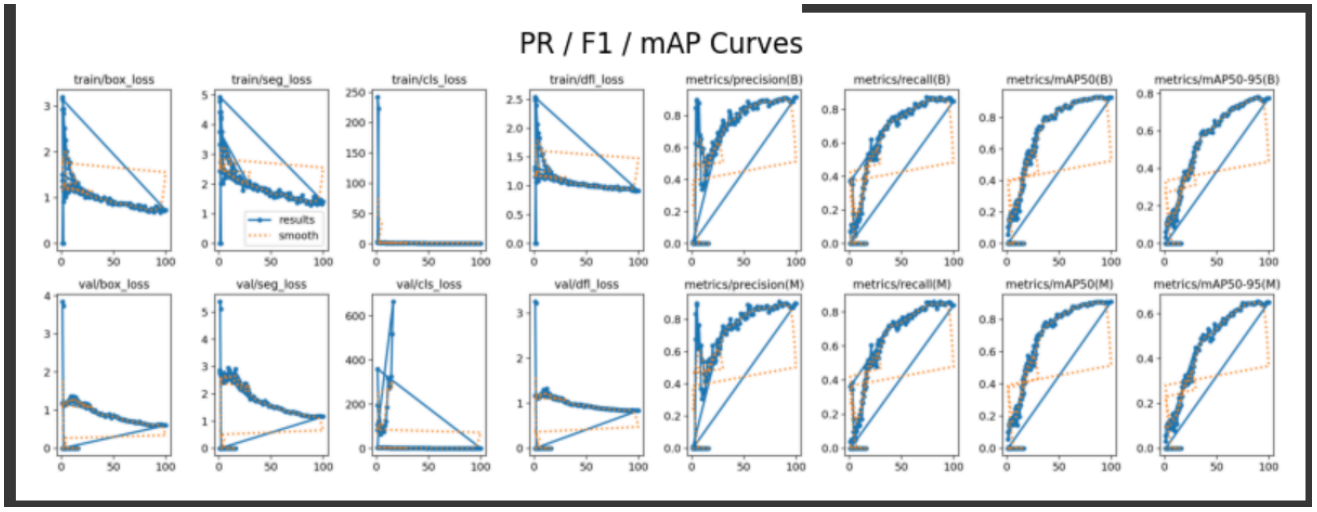
VALIDATION MATRIX


```
📊 Validation Metrics:
{
    "metrics/precision(B)": 0.4661728368929149,
    "metrics/recall(B)": 0.535887537792691,
    "metrics/mAP50(B)": 0.44489421286969044,
    "metrics/mAP50-95(B)": 0.29079342389425417,
    "metrics/precision(M)": 0.5099974046659622,
    "metrics/recall(M)": 0.4784613350752769,
    "metrics/mAP50(M)": 0.43647800623521926,
    "metrics/mAP50-95(M)": 0.28738832496792605,
    "fitness": 0.5781817488621802
}
```

CONFUSION MATRIX



PR /F1/MAP CURVE

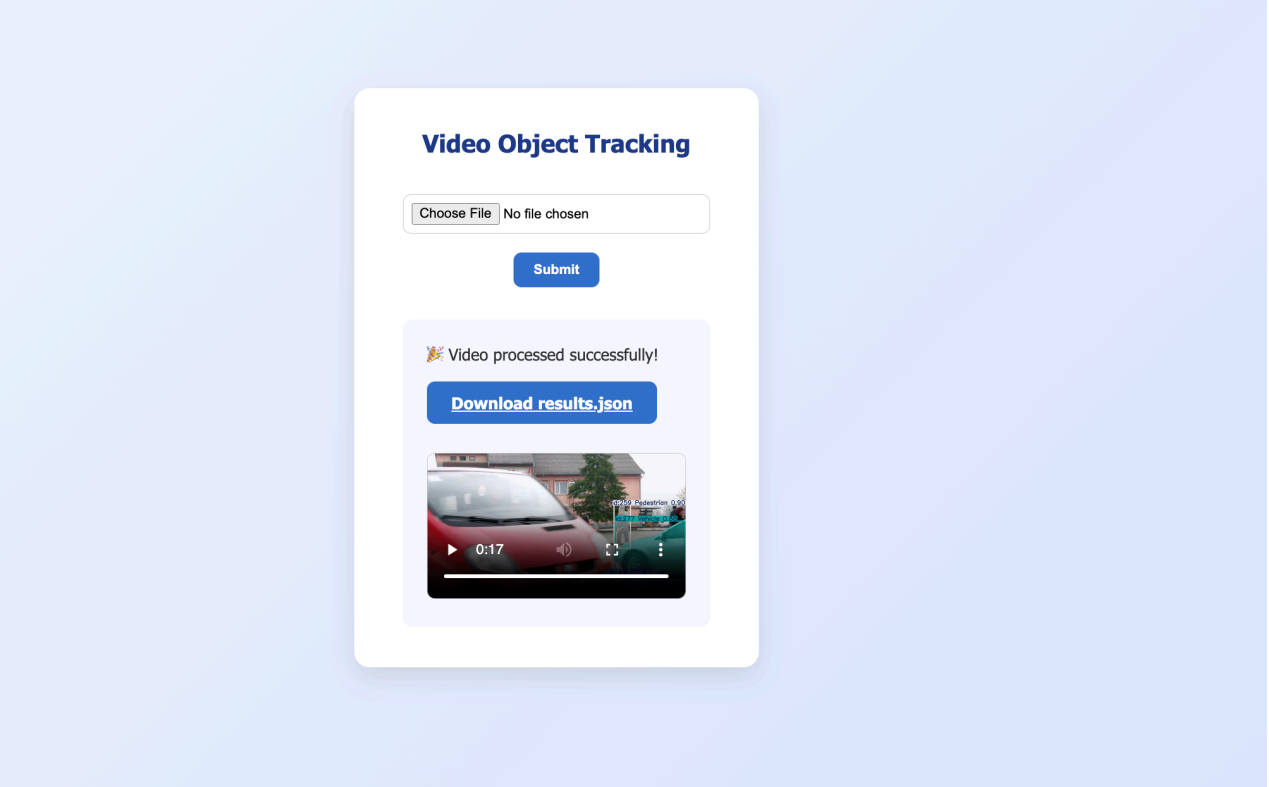


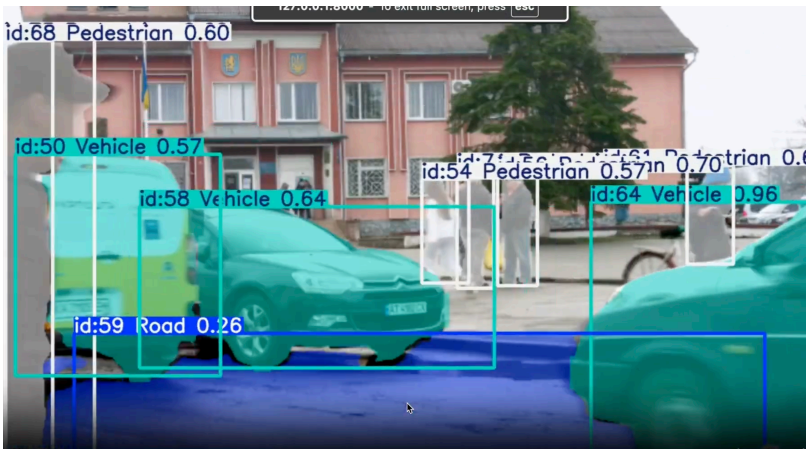
SUMMARY TABLE

```
Ultralytics 8.3.203 🚀 Python-3.12.11 torch-2.8.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
YOLOv8n-seg summary (fused): 85 layers, 3,258,844 parameters, 0 gradients, 11.3 GFLOPs
val: Fast image access ✅ (ping: 0.4±0.1 ms, read: 296.2±140.7 MB/s, size: 1037.1 KB)
val: Scanning /content/drive/MyDrive/campushiring/piyush_gupta/data/export-#Ug406mREPaSPGsWICLZw/labels/val.cache... 20 images, 3 backgrounds, 0 corrupt: 100% ━━━━━━━━━━━━ 20/20 37.0Kit/s 0.0s
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 2/2 0.3it/s 6.6s
                   all         20        115      0.466      0.536      0.445      0.291       0.51      0.478      0.436      0.287
                  Road         13         19      0.439      0.618      0.567      0.396      0.487      0.526      0.544      0.449
            TwoWheeler          4          9       0.27      0.451      0.294      0.176      0.309      0.444      0.294      0.188
            Pedestrian         14         47      0.547      0.574      0.441      0.234      0.546      0.468      0.444      0.205
               Vehicle         12         40      0.609        0.5      0.478      0.357      0.697      0.475      0.464      0.308
Speed: 9.3ms preprocess, 28.3ms inference, 0.0ms loss, 4.9ms postprocess per image
Results saved to /content/drive/MyDrive/campushiring/piyush_gupta/src/runs/yolo-seg-finetune
```

6. Video Tracking Demo

The Django web application enables users to upload videos and perform object detection and tracking using YOLOv8-Seg combined with ByteTrack. It processes each video to generate a structured results.json file containing tracking details such as object IDs, classes, and bounding boxes, while also providing visualizations with annotated bounding boxes and segmentation masks for easy interpretation. Designed with a user-friendly interface and scalable backend, the app makes advanced video analytics accessible and sets the foundation for applications in traffic monitoring, surveillance, and intelligent transportation, with potential future improvements including real-time streaming, additional object classes, and advanced analytics dashboards.





7. Workflow Summary

8. Learnings

- Learn about YOLO tracking configuration and troubleshooting methods.
- Solve type_track error by checking dataset format, class mapping, and pipeline compatibility apply fixes and verify track type usage.
- Monitor and resolve Colab GPU connection loss by reconnecting sessions, saving checkpoints, and investigating runtime limits for stability.

9. Conclusion

The development of a vehicle and pedestrian tracking system using YOLOv8-Seg in combination with ByteTrack demonstrates the effectiveness of modern deep learning models for robust, real-time multi-object tracking. The integration of segmentation-based detection with a strong tracking algorithm enabled accurate identification and continuous monitoring of objects across frames, highlighting the system's potential for applications in traffic monitoring, surveillance, and intelligent transport systems.

Future improvements can further enhance the robustness and scalability of this work. Training on larger and more diverse datasets will help the model generalize to a wider variety of urban environments and weather conditions. Real-time optimization, including model compression techniques and GPU/edge hardware acceleration, will allow deployment in resource-constrained systems such as roadside cameras or embedded automotive platforms. Lastly, expanding the object classes beyond vehicles and pedestrians to include bicycles, traffic signs, and road hazards will make the system more comprehensive and useful for advanced driver-assistance systems (ADAS) and smart-city infrastructure.