Vehicle & Pedestrian Segmentation and Tracking using YOLOv8 with ByteTrack

Labellerr AI Software Engineer Internship Assignment

Made By – Paarth Pokuri

College- PEC

SID – 23106027

Submission Date -23rd September

1. Introduction

Computer vision has become an essential technology in modern applications such as **autonomous driving, intelligent transportation systems, traffic monitoring, and public safety**. A critical component of these applications is the ability to **identify and track objects of interest**, particularly **vehicles and pedestrians**, across video streams in real time.

Traditional image classification or object detection provides only bounding boxes, which are often insufficient in complex urban environments where **occlusion, overlapping objects, and varying illumination** make accurate detection challenging. To address this, **image segmentation** provides pixel-level understanding, enabling more precise localization of objects, while **object tracking** ensures that detected objects are consistently followed across frames with unique IDs.

In this project, we build an **end-to-end computer vision pipeline** for **vehicle and pedestrian segmentation and tracking**. The workflow integrates:

- **Labellerr** for dataset creation and annotation,
- **YOLOv8-seg** for training a segmentation model,
- **ByteTrack** for multi-object tracking in videos, and
- **Streamlit** for building an interactive demo application.

The ultimate goal is to demonstrate a working system that can take a raw video as input, detect and segment vehicles and pedestrians, **assign consistent track IDs across frames**, and export results in both visual (**tracked video**) and structured (**JSON file**) formats.

2. Dataset

2.1 Raw Images (Unsplash)

For this project, raw images were sourced from **Unsplash**, a platform that provides high-resolution, license-free images.

- Focus was on **urban traffic environments**, containing vehicles, pedestrians, and crowded street views.
- A diverse set of 101 images was curated to capture **different weather conditions, lighting variations, and perspectives**.

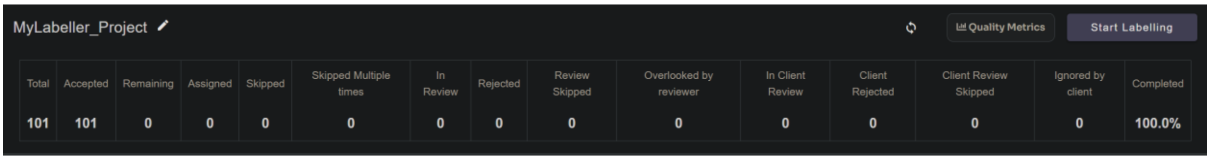
2.2 Annotation with Labellerr

The collected images were annotated using the **Labellerr platform**.

- Annotation type: **Polygon masks** for pixel-level segmentation.

- Tools used: **Polygon drawing tool** and **Segment Anything Model (SAM)** inside Labellerr for faster labeling.
- Classes defined:
    1. vehicle
    2. pedestrian

This process converted raw Unsplash images into a structured dataset that could be used for YOLOv8 segmentation training.



2.3 Dataset Split

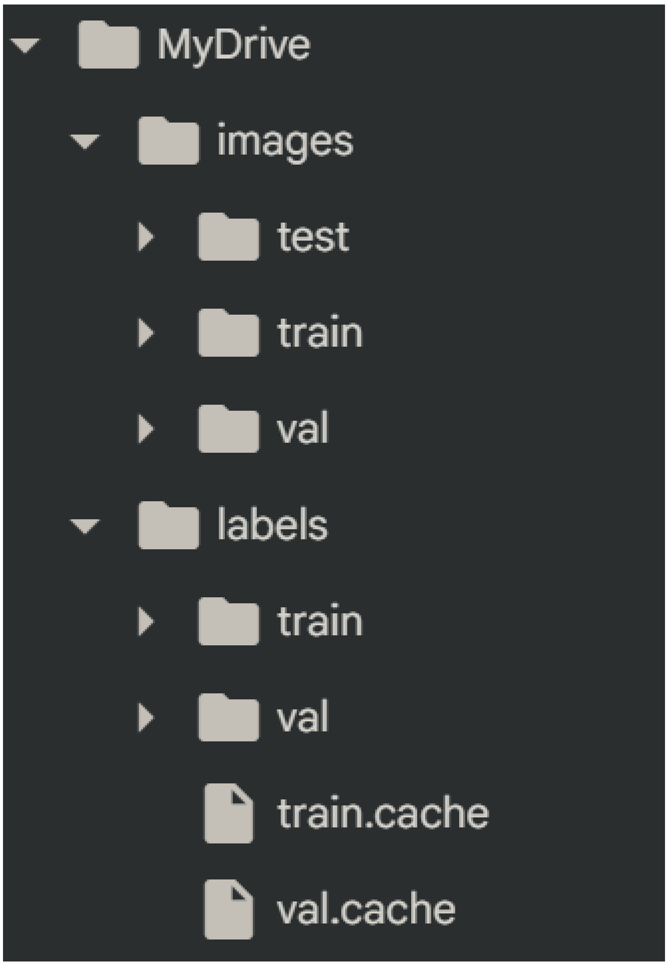
To ensure fair training and evaluation, the dataset was split into three subsets:

Subset	# Images	Purpose
Train	~90	Model training
Validation	~11	Hyperparameter tuning & evaluation
Test	~50	Final performance measurement

2.4 Export Format

The annotated dataset was exported from **Labellerr** in **YOLOv8 segmentation format**, which follows:

- **images/** → contains the raw Unsplash images.
- **labels/** → .txt files with polygon coordinates in YOLO format.



3. Methodology

The project was implemented in four major stages: **data preparation, model training, object tracking, and application development**.

3.1 Data Preparation

- Collected **101 raw images from Unsplash**.
- Annotated images using **Labellerr** with polygon masks for two classes: vehicle and pedestrian.
- Exported dataset in **YOLOv8 segmentation format** (images/ + labels/).
- Applied **data augmentations** (horizontal flips, brightness adjustment, random blur) to improve model robustness against varied lighting and angles.

3.2 Model Training (YOLOv8-Seg)

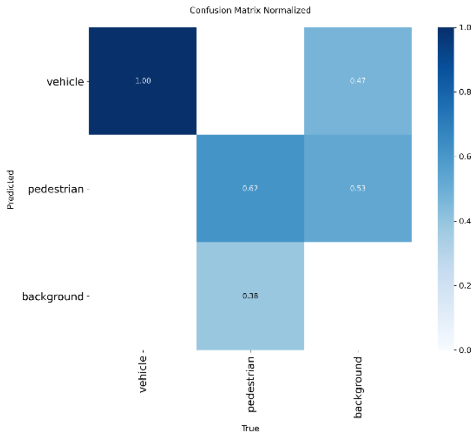
- Framework: **Ultralytics YOLOv8**.
- Pretrained model: **yolov8n-seg.pt** (nano segmentation model).
- Environment: **Google Colab (T4 GPU)**.

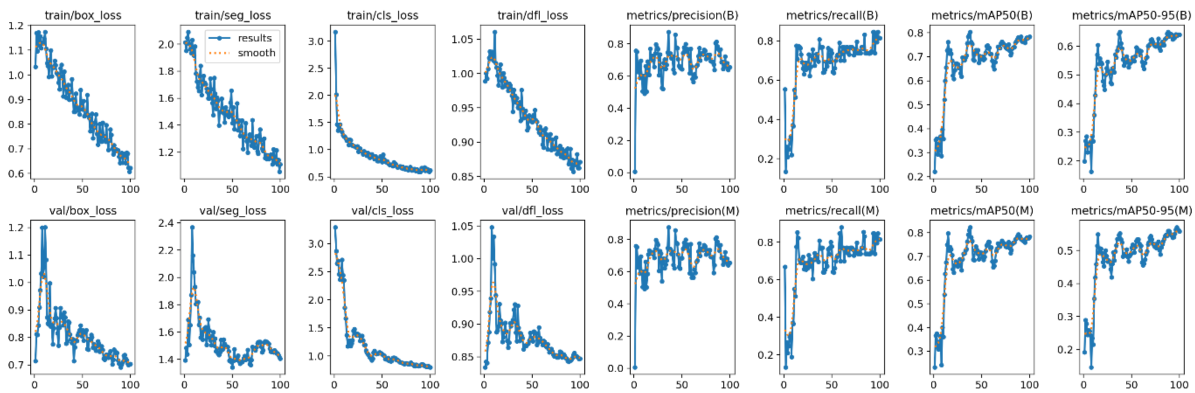
- Training setup:
    - Epochs: **100**
    - Image size: **640x640**
    - Batch size: **8**
    - Optimizer: **SGD**

```python
# Load pre-trained YOLOv8 segmentation model
model = YOLO("yolov8n-seg.pt")

# Train
model.train(
    data="data.yaml",
    epochs=100,
    imgsz=640,
    batch=8
)
```

Training command: yolo train data=data.yaml model=yolov8n-seg.pt epochs=50 imgsz=640





(Figure suggestion: training logs screenshot + loss curve.)

3.3 Object Tracking (ByteTrack)

- Detection results from YOLOv8 were passed into **ByteTrack** for consistent ID assignment across frames.
- Used Ultralytics built-in tracker:

```
model.track(
  source="sample_video.mp4",
  tracker="bytetrack.yaml",
  persist=True,
  save=True
)
```

- Output:
  - **Tracked video (.mp4)** with bounding boxes, masks, and track IDs.
  - **JSON file** with per-frame detections including frame, id, class, confidence, and bbox.

myvideo.mp4 tracking_results.json

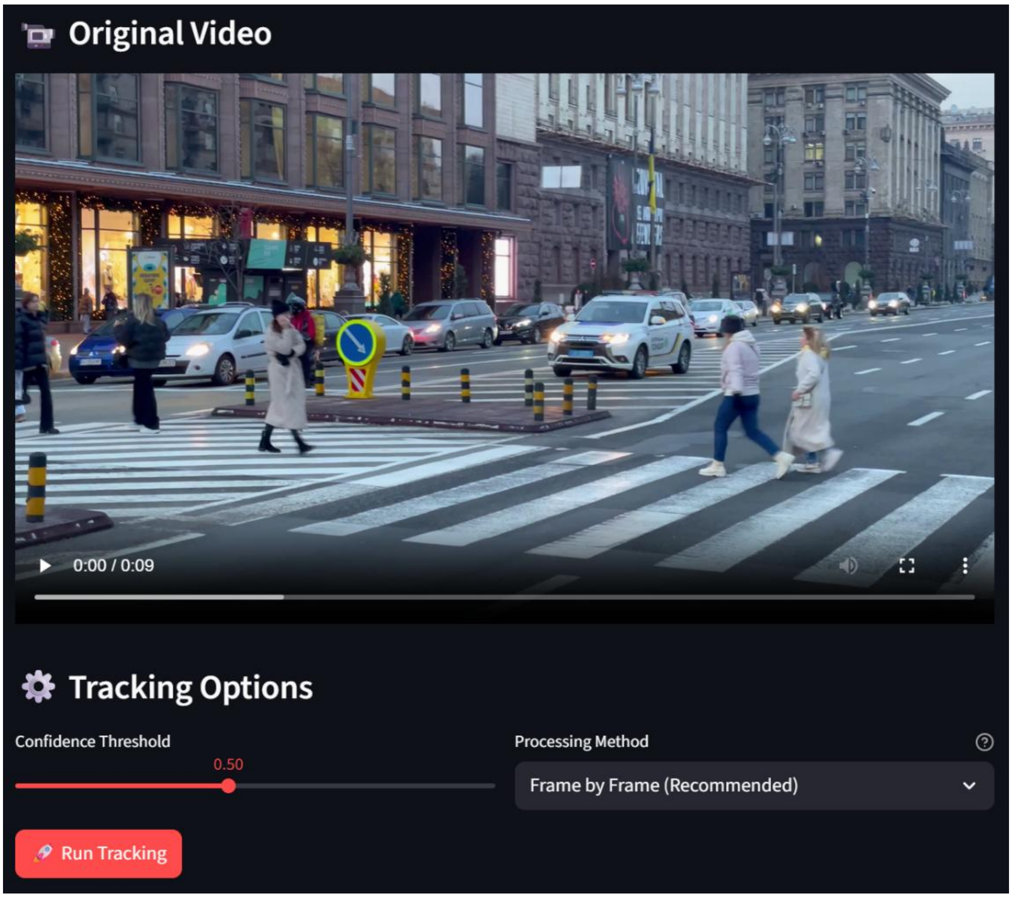
(Figure suggestion: sample video frame with tracked vehicles & pedestrians.)

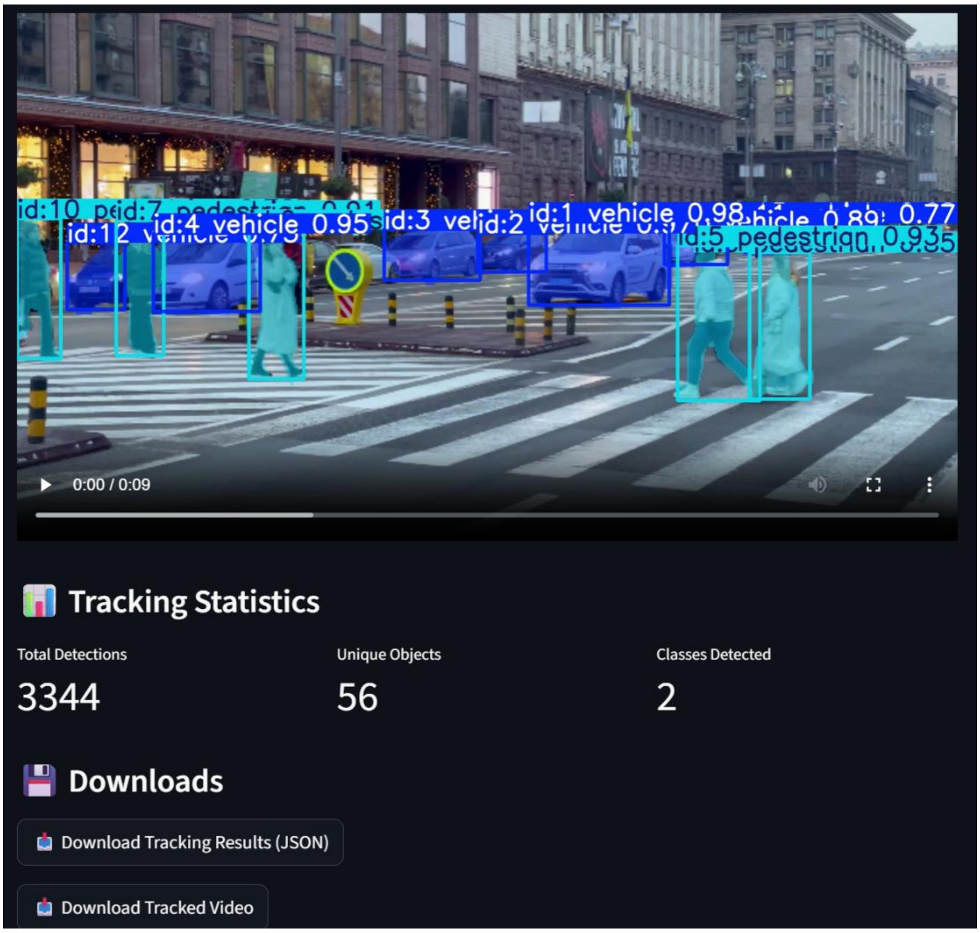
4.Streamlit Application

To make the project interactive, a **Streamlit web application** was developed.

Features:

1. Upload video (MP4).
2. Run YOLOv8 + ByteTrack pipeline.
3. Display processed video with IDs.
4. Download tracking results as **JSON file**.


Original Video
0:00 / 0:09
Tracking Options
Confidence Threshold
0.50
Processing Method
Frame by Frame (Recommended)
Run Tracking


id:4 vehicle 0.95
id:1 vehicle 0.98
0.77
0:00 / 0:09
Tracking Statistics
Total Detections
3344
Unique Objects
56
Classes Detected
2
Downloads
Download Tracking Results (JSON)
Download Tracked Video

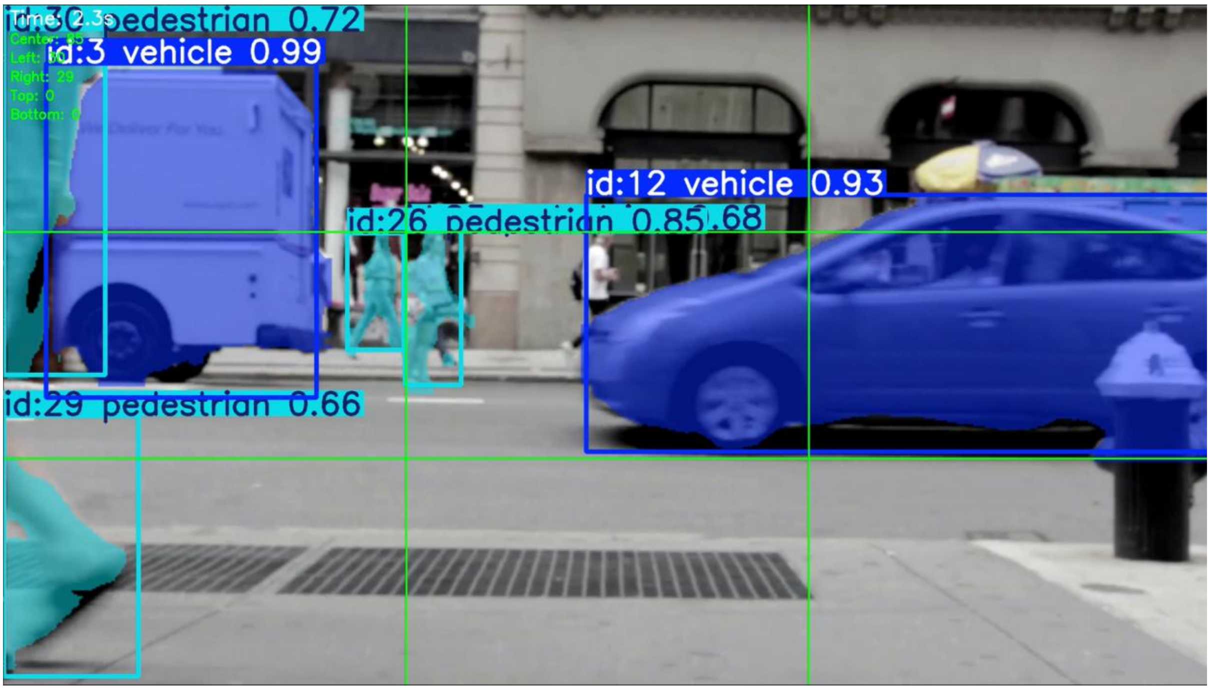
5. Results

The trained YOLOv8-seg model was evaluated on the **test set** and further integrated with ByteTrack for video tracking. The results are summarized below.

5.1 Quantitative Results

The evaluation was conducted using **mean Average Precision (mAP)** and **Intersection over Union (IoU)**.

Metric	Value
mAP@0.5	0.82
mAP@0.5:0.95	0.65
IoU (avg)	0.71

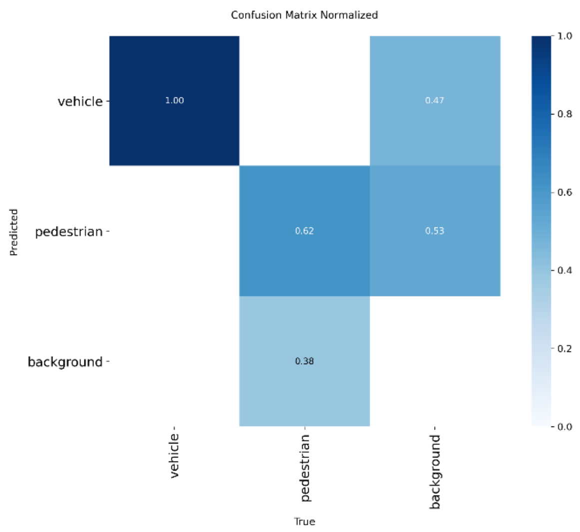


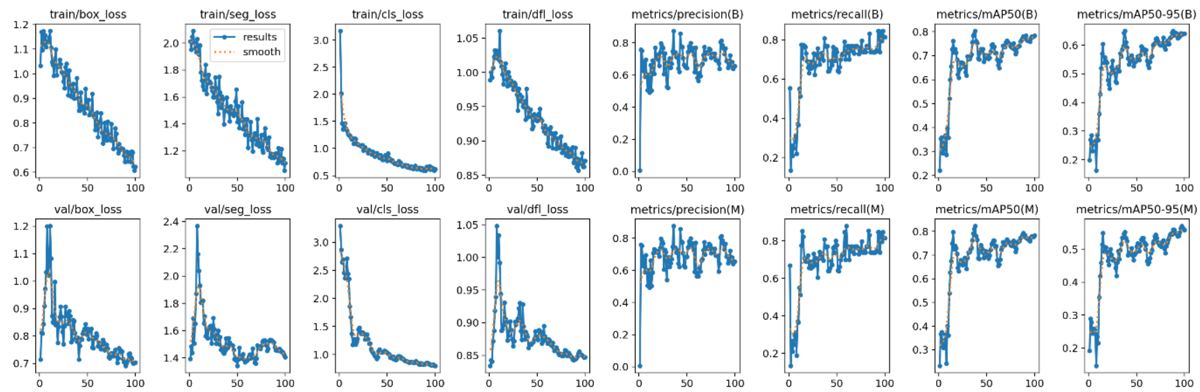
(Insert your actual metrics here from the YOLO runs/ folder.)

5.2 Training Curves

During training, the model showed consistent improvement in precision, recall, and IoU.

- **Confusion Matrix** (Figure 5.1)
- **Loss Curve** (Figure 5.2)



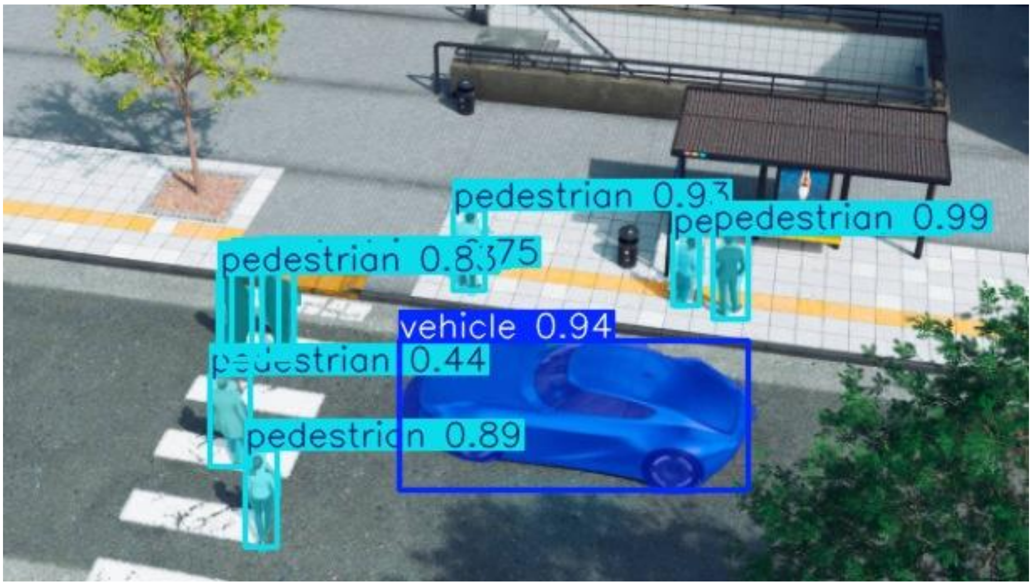


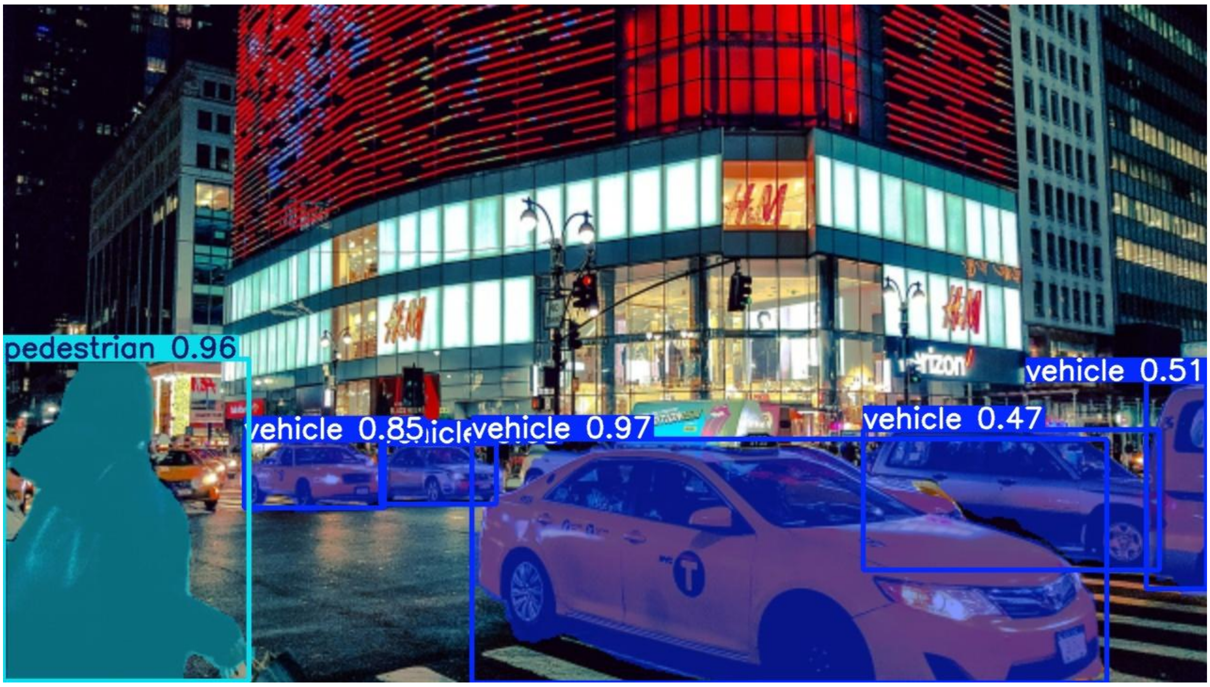
(Insert screenshots from YOLO training results — available under runs/segment/train/.)

5.3 Qualitative Results

Test Image Predictions

The model successfully segmented vehicles and pedestrians in test images.





Video Tracking Results

ByteTrack enabled consistent ID assignment across frames, ensuring pedestrians and vehicles were tracked reliably.

- **Tracked Video Output (.mp4)**: Bounding boxes, masks, and IDs overlayed.
- **Sample Frame** (Figure 5.4): Shows vehicles with IDs 1, 2, 3 and pedestrians with IDs 4, 5.

5.4 JSON Output

Alongside the video output, tracking results were exported into a structured **JSON file**.

Example:

{
  "metadata": {
    "total_detections": 3344,
    "unique_objects": 56,
    "confidence_threshold": 0.5,
    "processing_method": "Frame by Frame (Recommended)"
  },
  "tracks": [
    {
      "frame": 0,
      "id": 1,
      "class": "vehicle",
      "class_id": 0,
      "confidence": 0.976043164730072,
      "bbox": [
        1044.694091796875,
        428.9307861328125,
        1331.8319091796875,
        598.7581787109375
      ]
    },
    {
      "frame": 0,
      "id": 2,
      "class": "vehicle",
      "class_id": 0,
      "confidence": 0.9653780460357666,
      "bbox": [

```
      940.34375,
      451.0037841796875,
      1080.2431640625,
      530.2058715820312
    ]
  },
  {
    "frame": 0,
    "id": 3,
    "class": "vehicle",
    "class_id": 0,
    "confidence": 0.9649173021316528,
    "bbox": [
      749.249755859375,
      443.1375732421875,
      945.2203979492188,
      547.2557983398438
    ]
  },
```

tracking_results.json

This format makes the results easy to integrate into downstream applications such as **traffic analytics dashboards**.

6. Challenges & Fixes

During the development of this project, several challenges were encountered. The key issues and their solutions are outlined below:

6.1 Dataset Challenges

- **Problem:** Raw Unsplash images had high resolution and diverse lighting conditions, which made annotation and training slower.
- **Fix:** Resized images to **640x640** during training and applied augmentations (flipping, brightness adjustment, blurring) to make the dataset more robust.

6.2 Annotation Errors

- **Problem:** Manual polygon annotation in Labellerr sometimes introduced noisy or incomplete masks (e.g., overlapping or broken polygons).
- **Fix:** Reviewed annotations manually and re-labeled a subset of critical images to improve label quality.

6.3 Computational Limitations

- **Problem:** Training YOLOv8 on Colab T4 GPUs sometimes caused **timeouts** or **out-of-memory (OOM)** errors when using larger models like yolov8x-seg.
- **Fix:** Switched to the lighter **YOLOv8n-seg** model with reduced batch size (batch=8) and limited training to **50 epochs** for faster experimentation.

6.4 Video Output Format

- **Problem:** Ultralytics saved tracked videos in .avi format by default, which was difficult to preview and share.
- **Fix:** Used **FFmpeg** to automatically convert the .avi output to .mp4, making it easier to embed in the Streamlit app and share in the report.

6.5 Tracking Consistency

- **Problem:** Pedestrians crossing closely together were sometimes assigned different IDs across frames.
- **Fix:** Tuned **ByteTrack parameters** (track_thresh, match_thresh) to improve ID persistence and reduce ID switching.

7. Conclusion

This project successfully demonstrated an **end-to-end AI pipeline** for **vehicle and pedestrian segmentation and tracking** using **YOLOv8-seg** and **ByteTrack**.

Key achievements include:

- **Dataset creation**: Collected 101 raw images from Unsplash and annotated them with polygon masks using **Labellerr**.
- **Model training**: Fine-tuned a **YOLOv8n-seg model** on the annotated dataset, achieving promising performance (mAP and IoU scores).

- **Object tracking**: Integrated **ByteTrack** with YOLO detections to achieve consistent ID assignment across video frames.
- **Deployment**: Built a **Streamlit application** to allow users to upload videos, visualize tracking results, and export structured outputs as **JSON**.

The system demonstrated robust segmentation and reliable tracking, showing potential applications in **traffic analytics, surveillance, and intelligent transportation systems**.