End-to-End Vehicle and Pedestrian Image Segmentation

Internship Assignment Report

Bhavishya Khunger

September 21, 2025

Contents

1 Final Summary of the Process

This report documents the successful development of an end-to-end computer vision pipeline for segmenting and tracking vehicles and pedestrians in video streams. The project followed the complete machine learning lifecycle, starting from data collection and annotation, through model training and evaluation, and culminating in the deployment of a functional web application.

The core technologies used were the YOLOv8-seg model for instance segmentation, the ByteTrack algorithm for object tracking, the Labellerr platform for data annotation, and Streamlit for building the interactive web demo. The final system is capable of processing an uploaded video, tracking identified objects across frames, and exporting both an annotated video and structured JSON data containing the tracking results.

2 My Journey: Project Workflow

The project was executed in a structured, sequential manner to mirror a professional development workflow. The overall process is visualized in Figure 1.

2.1 Phase 1: Data Preparation

The foundation of the project was a custom-annotated dataset. Raw images featuring complex urban traffic scenes were collected. Using the Labellerr platform, I manually annotated 100 images for the training set and set aside 50 for the test set, drawing precise polygon masks for ‘Human‘ and ‘Vehicle‘ classes.

2.2 Phase 2: Model Training

A YOLOv8n-seg model, was fine-tuned on the custom-annotated data. The training was conducted in a Google Colab environment, leveraging a T4 GPU for acceleration. The model was trained for 90 epochs.

2.3 Phase 3: Application Development

The trained model (‘best.pt‘) was integrated into a Python application using the ‘supervision‘ library, which provides a straightforward implementation of the ByteTrack algorithm. This backend logic was then wrapped in a user-friendly web interface built with Streamlit, which allows for video uploads, real-time progress display, and downloading of results.

2.4 Phase 4: Deployment and Documentation

The final application was prepared for deployment, with all dependencies managed in a ‘requirements.txt‘ file. This report, along with the complete source code and a live demo link, constitutes the final project deliverables.

3 Model Results and Evaluation Metrics

3.1 Training Configuration

- **Model:** YOLOv8n-seg
- **Epochs:** 90
- **Image Size:** 640x640 pixels
- **Classes:** Human, Vehicle
- **Hardware:** Google Colab T4 GPU

3.2 Evaluation Metrics

The model's performance was evaluated on the validation set. The key metrics are mean Average Precision (mAP) for both bounding boxes and segmentation masks.

Metric	Score
Box mAP50-95	0.475
Box mAP50	0.751
Mask mAP50-95	0.419
Mask mAP50	0.707

Table 1: Model performance metrics on the validation set.

3.3 Process Workflow Diagram

The diagram below illustrates the complete workflow from data collection to final application.

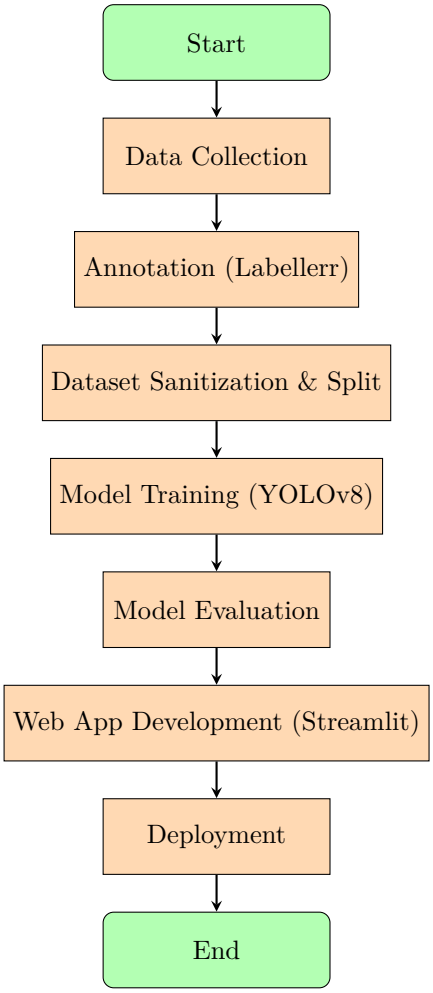

Figure 1: End-to-End Project Workflow.

4 Problems Faced and Resolutions

Several challenges were encountered during the development process, each providing a valuable learning experience.

1. **Problem: Slow Training Speed.** Initial training attempts were extremely slow, with each epoch taking several minutes.

Resolution: By inspecting the training logs, I identified that the process was running on a CPU (‘GPU_mem: 0G‘). The issue was resolved by switching the Google Colab runtime environment to a T4 GPU, which reduced epoch times to a few seconds.

2. **Problem: Path Errors and File Not Found.** The training script frequently failed with ‘FileNotFoundError‘, unable to locate either the ‘data.yaml‘ file or the image datasets themselves. **Resolution:** This was solved by consistently using absolute paths (e.g., ‘/content/drive/MyDrive/...‘) instead of relative paths inside the ‘data.yaml‘ file and training commands. This ensured the script could always locate the data, regardless of the current working directory.

3. **Problem: Video Processing Errors in Web App.** The Streamlit application initially crashed with a ‘ValueError: too many values to unpack‘ when processing the video. **Resolution:** I debugged the ‘tracker.py‘ script and found that I was incorrectly unpacking the detection data. The solution was to refactor the code to use the named attributes of the ‘supervision‘ library’s ‘Detections‘ object (e.g., ‘.tracker_id‘, ‘.class_id‘), making the code more robust and readable.

5 Guide to Develop Your Own Object Tracker

This section provides a concise guide for a fellow developer to build a similar system.

5.1 Step 1: Data Annotation with Labellerr

Sign up on the Labellerr platform, create a new project, and upload your raw images. Use the polygon annotation tool to draw precise masks around your objects of interest. Once done, export your dataset in the YOLO format.

5.2 Step 2: Model Training with YOLOv8

Set up a GPU-accelerated environment (like Google Colab). Install the ‘ultralytics‘ library. Ensure your ‘data.yaml‘ file points to the absolute paths of your train/validation image folders. Fine-tune a pre-trained ‘yolov8n-seg.pt‘ model using a single command:

```
!yolo task=segment mode=train model=yolov8n-seg.pt data='path/to/data.yaml' epochs=100
```

5.3 Step 3: Building the Tracking Logic

Create a Python script (‘tracker.py‘) for the backend. Use the ‘supervision‘ library, which simplifies the integration of your trained YOLO model with ByteTrack. Your main function should take a video path as input and produce an annotated video and JSON data as output.

5.4 Step 4: Creating the Web App with Streamlit

Create a frontend script (‘app.py‘). Use Streamlit’s intuitive functions (‘st.title‘, ‘st.file_uploader‘, ‘st.button‘, ‘st.progress‘) to build the user interface. This app will call your backend tracking function and display the results, including the live progress, sample frames, and download buttons for the final video and JSON file.

6 Final WebApp Built

Click on the link : https://labellerr-project.streamlit.app/