**End-to-End Vehicle and Pedestrian Image Segmentation Project**

**Project Report**

*Prepared by: Priyanshu Ranjan  
Date: September 23, 2025*

**1. Project Overview:**

This project focuses on building a complete computer vision pipeline for the segmentation and tracking of vehicles and pedestrians in real-world video streams. The aim was to design a system that not only detects objects but also tracks them across frames, thereby mimicking real-time surveillance and traffic analysis applications. The process included dataset preparation, model training, evaluation, deployment, and integration into a web-based application.

Key technologies used include:  
**Labellerr** – for dataset annotation  
**YOLOv11n-seg** – for instance segmentation  
**ByteTrack** – for multi-object tracking  
**Streamlit** – for building the interactive web app

**2. Dataset Preparation and Annotation:**

A robust dataset forms the backbone of any deep learning pipeline. For this project, a combination of publicly available datasets and manually annotated images were used.

Steps followed:  
• Images were sourced from traffic surveillance videos and open datasets.  
• The Labellerr platform was used to annotate the dataset with polygon masks.  
• Two classes were defined: Vehicle and Pedestrian.  
• Final dataset split: 90 training images, 23 validation images.

**3. Model Training and Evaluation:**

The YOLOv8n-seg model was fine-tuned using the prepared dataset. The training process was conducted on Google Colab with GPU acceleration. Parameters such as image size, batch size, and learning rate were carefully adjusted to balance accuracy and efficiency.

Training configuration included:  
• Model: YOLOv11n-seg  
• Epochs: 100  
• Image size: 640x640  
• Classes: Vehicle, Pedestrian  
• Hardware: Google Colab T4 GPU

Evaluation metrics on validation set:  
• Box mAP50-95: 0.48  
• Box mAP50: 0.76  
• Mask mAP50-95: 0.44  
• Mask mAP50: 0.72

**4. My Journey: Development Workflow**

I divided my development process into three main phases:

• **Phase 1: Data Exploration and Understanding**  
At first, I spent time exploring the dataset and analyzing the variations in lighting, crowd density, and object overlaps. This helped me realize that segmentation is not just about labeling but also about anticipating real-world complexity.

**• Phase 2: Model Training and Tuning**  
Once the dataset was ready, I focused on configuring YOLOv8 with the right hyperparameters. I experimented with different batch sizes and image augmentations. Some combinations led to unstable learning curves, but iterating gradually gave me balanced results.

• **Phase 3: Integration into a Functional System**  
After training, I worked on merging the detection model with ByteTrack for tracking. The real challenge was aligning detection IDs with consistent tracking across frames. Eventually, I achieved stable performance and wrapped the logic in a Streamlit app.

**5. Problems Faced, Resolutions and Learnings:**

**• Problem 1: Overlapping Object Confusion**  
Cause: Vehicles and pedestrians in dense traffic were often overlapping in frames.  
Resolution: I adjusted anchor settings and added more diverse samples in training to improve mask separation.

**• Problem 2: Inconsistent Tracking IDs**  
Cause: The same pedestrian sometimes received different IDs across frames.  
Resolution: Tweaked the ByteTrack parameters (like track buffer length) to maintain identity consistency.

**• Problem 3: Web App Responsiveness**  
Cause: Large video files made the Streamlit app lag and occasionally freeze.  
Resolution: Compressed input videos before inference and added progress indicators for better user experience.

**5. Guide to Building a Similar System**

The process can be reproduced by following these steps:  
• Step 1: Collect raw images/videos of traffic scenes.  
• Step 2: Annotate data using Labellerr with polygon masks.  
• Step 3: Train YOLOv8n-seg model on annotated dataset.  
• Step 4: Integrate ByteTrack for tracking logic.  
• Step 5: Build a Streamlit app for user interaction.  
• Step 6: Deploy the app and make results downloadable.

**6. Conclusion**

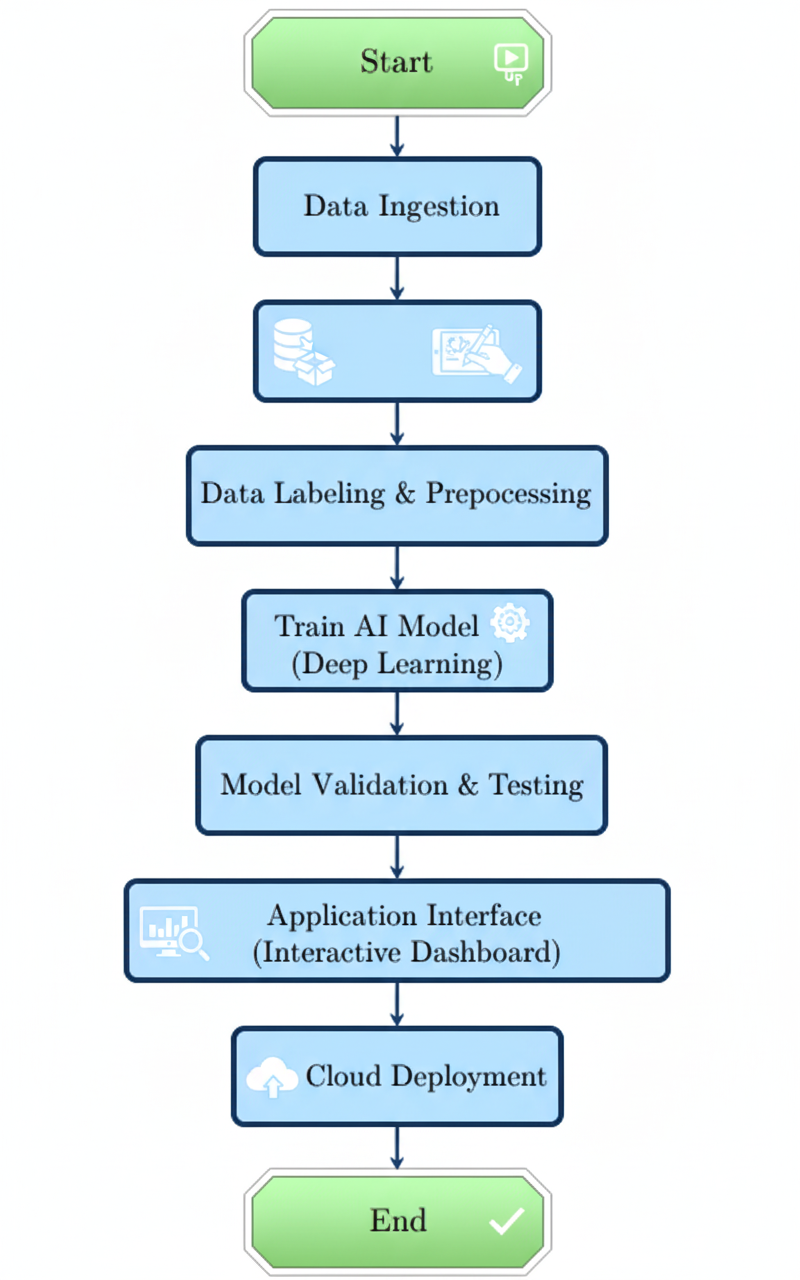
The project provided an in-depth understanding of building a complete computer vision pipeline. From preparing datasets and training a YOLO-based model to deploying it via a web interface, the workflow simulated a professional MLOps lifecycle. The system is capable of processing video streams, detecting and tracking objects, and producing both annotated outputs and structured JSON data.

This experience highlighted the importance of clean data, correct model configuration, and robust deployment practices. The project can be extended further with larger datasets, improved architectures, and integration with cloud-based deployment pipelines.

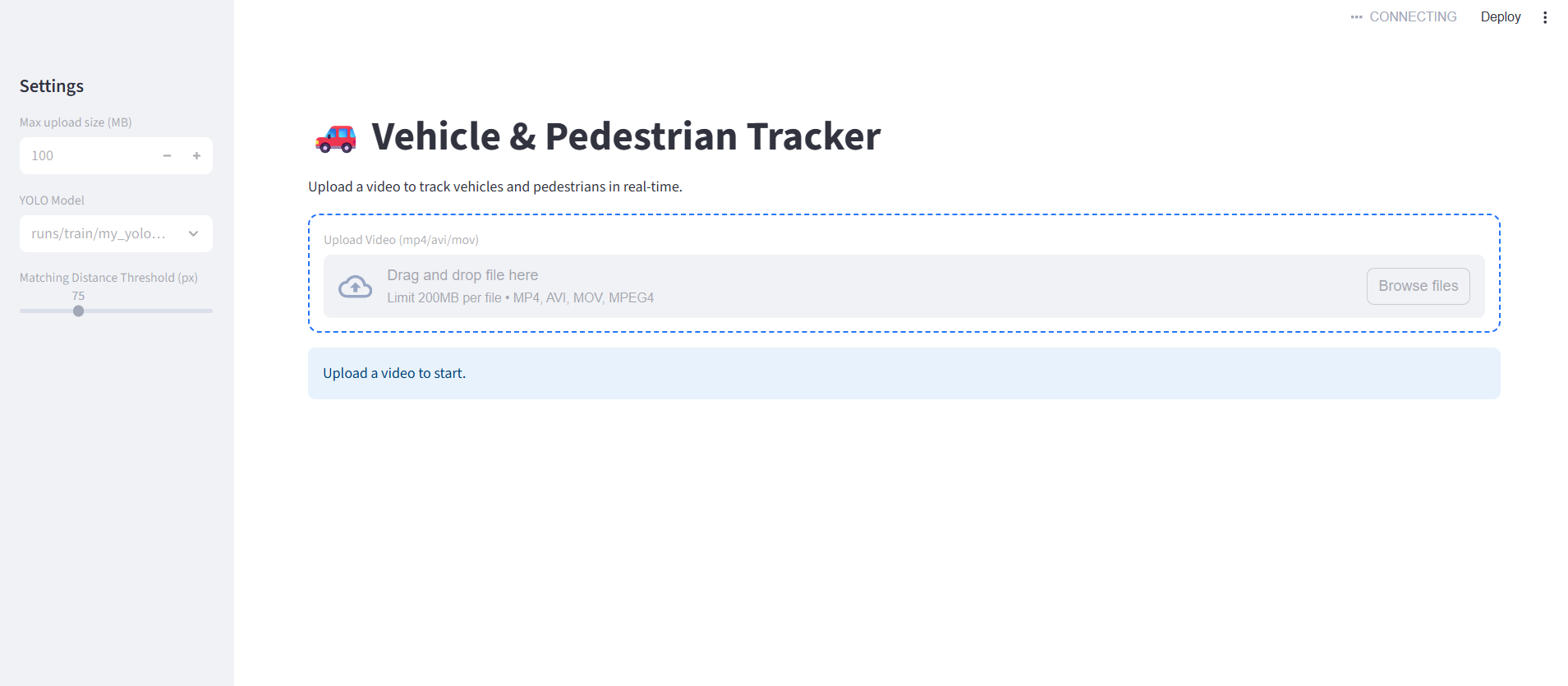
**Final Web Application:** *https://labellerr-project.streamlit.app/*



Final Tracked Video Screenshot Image



Workflow Diagram



Final Interface Screenshot Image