Traffic Detection System using YOLOv8

1. Project Approach

This project focuses on detecting and counting vehicles in images and videos using a deep learning model. The complete system is implemented using YOLOv8 for object detection, OpenCV for image processing, and Streamlit for the frontend interface. It enables users to upload media, process them using YOLOv8, and view/download annotated results along with analytical reports.

2. Model Selection

The YOLOv8 model (You Only Look Once version 8) from Ultralytics was chosen for its balance between speed and accuracy. It's ideal for real-time applications and easy to deploy. Only the following vehicle-related classes were used from the COCO dataset:
- Class ID 2: Car
- Class ID 3: Motorcycle
- Class ID 5: Bus
- Class ID 7: Truck

A confidence threshold of 0.5 is set to minimize false detections.

3. Implementation Details

- The main detection logic resides in detector.py, which initializes the YOLO model and processes each image or video frame.

- The detect_vehicles function filters and annotates vehicles and overlays the count.

- Processed images are saved, and CSV reports are generated through utils.py.

- A Streamlit app (vehicle_app.py) allows users to upload images, view annotated results, and download outputs.

- The config.py file centralizes settings like directories, model path, class mapping, font settings, and bounding box colors.

4. Results Analysis

- The system accurately detects and classifies vehicles in both sparse and dense traffic images.
- Annotated images and video frames clearly show bounding boxes and count overlays.
- A CSV report is generated with vehicle counts for each image or video processed.
- Matplotlib-based bar graphs offer a clear visual representation of vehicle distribution.

Example Output (Image):

- Car: 4
- Bus: 1
- Motorcycle: 2

Example Output (Video):

- Total Cars: 78
- Total Trucks: 15
- Total Buses: 5

The annotated images/videos and reports validated the correctness of detection for multiple environments (urban roads, highways).

5. Challenges Faced

- Video frame processing caused occasional lag on lower-spec systems.
- Class misclassification occurred between similar vehicle types (e.g., trucks vs. buses).
- Streamlit temporary file handling needed careful cleanup to avoid memory issues.
- Ensuring consistent accuracy across varied lighting and image quality was a key concern.

6. Potential Improvements

- Fine-tune YOLOv8 on a dedicated traffic dataset for better precision.
- Integrate multi-object tracking to avoid double counting in videos.
- Extend the app to support real-time webcam or CCTV feed processing.
- Add location-based tagging and summary statistics for surveillance applications.