TECHNICAL REPORT

1. Objective

The goal of this project is to develop a computer vision system that can automatically detect, classify, and count vehicles (cars, motorcycles, trucks, and buses) in traffic images. This solution aims to support intelligent traffic monitoring, congestion analysis, and smart city applications by leveraging deep learning-based object detection techniques.

2. Approach

To build the vehicle detection pipeline, we used the following structured approach:

- **Image Ingestion:** Load a high-resolution traffic image from the web.
- **Model Selection:** Use a pre-trained YOLOv8m (medium) model for object detection with high accuracy and moderate inference time.
- **Inference & Filtering:** Detect objects using the model and filter detections to only include relevant vehicle classes based on the COCO dataset.
- **Post-processing:** Apply Non-Maximum Suppression (NMS) to refine bounding boxes and avoid multiple detections for the same object.
- **Annotation & Labeling:** Annotate the detected vehicles with class names and confidence scores using supervision utilities.
- **Counting & Summary:** Count each type of vehicle detected and overlay the count summary on the image with a transparent background.

3. Model Selection

- **Model Used:** YOLOv8m-1280 (YOLOv8 Medium with 1280x1280 resolution)
- **Why YOLOv8?** YOLOv8 provides state-of-the-art object detection performance with lightweight inference. The medium variant balances accuracy and speed.
- **Trained On:** COCO Dataset, which includes relevant classes such as car, motorcycle, bus, and truck.

4. Implementation Details

- **Language & Libraries:** Python with OpenCV, NumPy, and the supervision and inference libraries for easy integration and annotation.
- **Thresholds:** Detection confidence threshold was lowered to 0.3 to allow more inclusive detections. NMS threshold was set at 0.4 to suppress overlapping boxes.
- **Class Filtering:** Only COCO classes 2 (car), 3 (motorcycle), 5 (bus), and 7 (truck) were retained for analysis.
- **Visualization:** Annotated using bounding boxes and text overlays with adjustable font scale, padding, and transparency for better readability.

1. Results Analysis

Input Image

Output Image

(Example visualization with bounding boxes and count overlay)
[Annotated Output – vehicle boxes, labels, and count summary]
Detected Vehicles (example counts):

- Cars: 18
- Motorcycles: 2
- Trucks: 5
- Buses: 1

The YOLOv8m model effectively detected most vehicles in the image, even under cluttered traffic scenarios. Bounding boxes closely aligned with actual vehicle positions, and confidence scores were generally above 0.50.

2. Challenges Faced

- Overlapping Vehicles: Dense traffic led to overlapping bounding boxes, requiring fine-tuned NMS parameters to avoid false duplicates.
- Small Motorcycles: Smaller vehicles, like motorcycles far from the camera, were sometimes missed or misclassified.
- Shadow and Lighting Variations: Inconsistent lighting or shadows on roads occasionally impacted model confidence.
- Cloud Model Dependency: The inference module abstracts away the model internals, limiting custom training or optimization flexibility.

3. Potential Improvements

- Model Fine-Tuning: Train or fine-tune YOLOv8 on a custom dataset focused on Indian traffic images for better localization and class accuracy.
- Multi-Frame Analysis: Extend the system to process video or real-time feed to enable vehicle tracking and traffic flow estimation.
- Heatmap Generation: Add region-wise density maps to visualize congested zones in the traffic image.
- Edge Deployment: Optimize model for edge devices (e.g., Jetson Nano, Raspberry Pi) for real-time, on-site traffic monitoring.
- Expand Class Detection: Add more granular categories like auto-rickshaws, vans, or bicycles based on application needs.