# Vehicle Detection and Classification using YOLOv8

## 1. Objective

This project focuses on building an intelligent traffic monitoring system capable of detecting and classifying vehicles from visual data using the YOLOv8 deep learning model. The aim is to distinguish between two key vehicle categories:

- **Light Motor Vehicles (LMVs):** including cars, motorcycles, scooters, and autorickshaws.
- **Heavy Motor Vehicles (HMVs):** including trucks, buses, and tractors.

The system delivers both **visual output** (an annotated video stream with labeled detections) and **analytical output** (a CSV file logging vehicle counts per frame). This combination makes it suitable for a wide range of real-world applications, such as traffic analytics, urban planning, and intelligent transportation systems (ITS).

---

## 2. Tools and Technologies

The project leverages a robust set of technologies and frameworks:

- **Python:** Used as the primary language for its flexibility and ecosystem.
- **YOLOv8 (Ultralytics):** A state-of-the-art object detection model, known for high speed and accuracy, particularly effective in real-time applications.
- **OpenCV:** For video frame manipulation, annotation, and rendering.
- **ByteTrack:** To track vehicles across frames and avoid duplicate counts.

- **CSV Format:** Used for lightweight, structured data logging suitable for downstream statistical analysis.

---

## 3. Workflow Overview

#### 3.1 Model Loading

The system starts by loading a lightweight yet effective pre-trained YOLOv8-nano model. It balances detection performance and computational efficiency, especially on systems with limited GPU/CPU resources. Class names are dynamically extracted to support flexible categorization.

#### 3.2 Class Mapping

Detected objects are matched to either the LMV or HMV category. This is achieved by **normalizing class labels** (e.g., converting `"Motor bike"` to `"motorbike"`), ensuring robust classification even with minor name variations in the pretrained model.

#### 3.3 Input and Output Configuration

The system takes an input video file (typically `.mp4`) and generates:

- An **annotated output video** (`.avi`) showing bounding boxes, class labels, and track IDs.
- A **CSV file** that logs vehicle counts (LMVs and HMVs) for each frame.

To improve performance and maintain detection accuracy, the video is resized to 75% of its original resolution.

#### 3.4 Frame-wise Processing

Each video frame is handled through the following steps:

- **Detection:** YOLOv8 identifies vehicles within the frame.
- **Classification:** Each detected vehicle is labeled as LMV or HMV based on its class ID.

- **Annotation:** Bounding boxes and readable labels are drawn for user visualization.
- **Tracking:** ByteTrack assigns consistent IDs to detected vehicles across frames.
- **Logging:** The frame index and vehicle counts are saved into the CSV log.

#### 3.5 User Interaction

During runtime, the system displays the annotated video feed in real-time. The user can terminate processing at any point by pressing the `'q'` key.

---

## 4. Output Files

At the end of processing, the system outputs:

- **Annotated video:** With bounding boxes and classification info overlaid.
- **vehicle_counts.csv:** A frame-wise summary of LMVs and HMVs, enabling detailed offline analysis.

These files provide both **visual evidence** and **structured data**, ideal for reports, dashboards, and further ML modeling.

---

## 5. Key Features

- Real-time vehicle detection using **YOLOv8**
- Accurate LMV/HMV classification with **custom class groupings**
- **ByteTrack** integration to handle frame-to-frame identity tracking
- Lightweight design using **YOLOv8-nano** for higher FPS and lower memory usage

- Results stored in a **clean CSV format**, ready for aggregation, plotting, or database integration

- Configurable **confidence threshold** and **frame resizing** for speed-accuracy tuning

---

## 6. Results Summary: UA-DETRAC Subsample Evaluation

The system was tested on **8,379 images** extracted from the **UA-DETRAC** dataset to simulate a realistic traffic monitoring scenario. The key observations include:

- The system handled high vehicle density effectively, with **some frames containing up to 28 LMVs**.

- **LMVs** were significantly more frequent than HMVs, which aligns with typical urban traffic distributions.

- On average, each image had approximately **4–5 LMVs**, while HMVs appeared less frequently.

- The system consistently logged detections across a variety of scenes, showcasing good generalization across angles and lighting.

These results confirm the system's **reliability and scalability** in handling diverse traffic scenes in a structured format suitable for analysis.

---

## 7. Potential Improvements

The system can be extended in several impactful ways:

- **Enhance tracking robustness** by fusing motion models (e.g., Kalman filters) with ByteTrack

- **Zone-based vehicle counting**, such as entry/exit counts at specific regions of interest

- **Vehicle speed estimation**, using frame rate and object displacement across frames

- **Model retraining on local datasets** to improve detection for region-specific vehicle types or conditions

- **Real-time alerts or anomaly detection**, such as detecting traffic congestion or stalled vehicles

---

## 8. Limitations

While the system performs well in most cases, a few limitations exist:

- Detection accuracy may drop under **poor lighting**, **severe occlusion**, or with **rare vehicle types** not well represented in YOLOv8's training data.

- **Real-time performance** is dependent on hardware — particularly GPU availability and video resolution.

- **Bounding box overlap** in dense traffic may occasionally cause ID switching or duplicate detections.

---

## 9. Conclusion

This project demonstrates a functional and efficient approach to **vehicle detection and classification** using deep learning. By integrating **YOLOv8**, **OpenCV**, and **ByteTrack**, it delivers real-time visual analytics and structured outputs. The system serves as a valuable tool for **traffic pattern analysis**, **urban infrastructure planning**, and as a foundation for more complex smart transportation solutions.

The performance on the **UA-DETRAC dataset** further validates its robustness in real-world environments, setting the stage for practical deployment and further innovation.