TECHNICAL REPORT

#### An Assignment on Vehicle Detection and Counting model using YOLOv8

# INTRODUCTION

This project focuses on building a computer vision system to detect, classify, and count vehicles in traffic images using the state-of-the-art **YOLOv8** model. The system identifies **cars**, **trucks**, and **motorcycles** and provides visual feedback through bounding boxes and confidence scores.

# MODEL SELECTION

We used **YOLOv8**, the latest model in the YOLO family released by **Ultralytics**. YOLOv8 outperforms its predecessors in terms of speed, accuracy, and flexibility. It supports a PyTorch-based API with an intuitive inference process and is capable of detecting multiple object classes with higher precision, especially for smaller or occluded objects like motorcycles.

# TOOLS AND LIBRARIES

1. **Programming Language:** Python 3.8+
2. **Libraries Used:** OpenCV, NumPy, Matplotlib
3. **ML Framework:** YOLOv8 (via Ultralytics' Python package)
4. **Interface:** Jupyter Notebook

# IMPLEMENTATION PIPELINE

1. **Image Input:** Read test image dataset.
2. **Model Load:** YOLOv8 pre-trained on the COCO dataset.
3. **Inference:** Detections are obtained with bounding boxes, class IDs, and confidence scores.
4. **Filtering:** Only retain vehicle detections (car: 2, motorcycle: 3, truck: 7) with confidence > 0.5.
5. **Annotation:** Draw bounding boxes with labels and confidence scores.
6. **Counting:** Count vehicles by category per image.
7. **Output:** Save annotated images in output image folder.

# RESULTS

1. **Test Set:** 10 traffic images processed
2. **Detection Accuracy:** ~90% for clearly visible vehicles
3. **Precision on Small Vehicles:** YOLOv8 successfully detected distant motorcycles missed by YOLOv5 in prior experiments
4. **Processing Time:** Average < 1 second per image
5. **Example Counts:**
   1. Cars: 20
   2. Trucks: 5
   3. Motorcycles: 4

# CHALLENGES FACED

* **Image Resolution Variability:** Some images had low quality, making detections harder
* **Occlusions:** Partially hidden vehicles were inconsistently classified
* **Model Class IDs:** Custom filtering was needed to ensure only relevant vehicle classes were retained from YOLOv8 output

# Future Work

* Use **YOLOv8m or YOLOv8l** for improved accuracy on complex scenes
* Fine-tune YOLOv8 on a **custom labeled traffic dataset**
* Add **video support** and **real-time processing**
* Deploy a simple **web interface** using Flask or Streamlit
* Integrate **heatmap visualization** for traffic density

# CONCLUSION

The system effectively detects, classifies, and counts vehicles in traffic images using YOLOv8n, meeting the assignment goals. Despite minor issues like false positives, it performs well and provides a solid base for future improvements, including real-time support and a web interface.