Computer Vision Project

TRAFFIC DENSITY ESTIMATION

Team Members

Vasishtha Pandya (B22CS036)
Prajjwal Dixit (B22ES002)
Keshav Khandelwal (B22ES009)
Raj Vijayvargiya (B22AI067)
Raghavi Epuri (B22ES012)

Contents

Abstract

This project offers a comprehensive solution for estimating traffic density by integrating traditional Computer Vision (CV) techniques, such as background subtraction and contour detection, with advanced Deep Learning (DL) methods, including state-of-the-art object detection models like YOLO and SSD. The objective is to automate traffic analysis from real-time video feeds, enabling accurate detection and counting of vehicles across various conditions. The final solution is deployed through an interactive Streamlit web application, allowing users to easily upload video inputs and visualize traffic statistics in a user-friendly interface.

1 Introduction

Traffic congestion has become a pressing issue in modern urban areas, leading to increased travel time, fuel consumption, and environmental pollution. Efficient traffic monitoring and management are essential for addressing these challenges. This project aims to tackle the problem by leveraging a combination of traditional Computer Vision (CV) techniques and modern Deep Learning (DL) models to build a robust system for real-time traffic density estimation using surveillance footage.

The system is designed to automatically detect and count vehicles in video streams, providing accurate density estimates across various traffic conditions. To ensure accessibility and ease of use, the results are displayed through an interactive web interface built with Streamlit. This scalable and automated approach offers a practical solution for smart traffic monitoring, paving the way for data-driven decision-making in urban traffic management systems.

2 Objective

The primary objective of this project is to develop an efficient and scalable traffic density estimation system which is designed to:

- Accurately detect and count vehicles in real-time from video streams
- Categorize traffic conditions into distinct levels such as Low, Medium, and High
- Evaluate and compare the performance of various object detection algorithms
- Deliver results through an intuitive and interactive web-based interface using Streamlit

3 Approaches Used

3.1 Computer Vision-Based Approaches

Computer Vision (CV)-based approaches offer several practical advantages for traffic density estimation, particularly in scenarios where resources are limited. These methods are lightweight and computationally efficient, making them suitable for real-time deployment on devices with restricted processing power, such as embedded systems or edge devices. CV techniques do not require large annotated datasets or complex training procedures, significantly reducing development time and data dependency.

Their rule-based logic such as using background subtraction or frame differencing makes them more interpretable and easier to debug or fine-tune based on specific camera setups. Additionally, they exhibit low latency and can deliver prompt responses, which is crucial in traffic monitoring applications. In stable environments with fixed camera angles and consistent lighting, CV-based methods can achieve reasonably accurate results without the overhead of deep learning frameworks.

Two classical Computer Vision (CV) methods were implemented to estimate traffic density without the use of neural networks. These approaches are lightweight, relatively fast, and do not require training data.

3.1.1 Approach 1: Background Subtraction and Contour Detection

This approach involves preprocessing each frame and then applying background subtraction using algorithms like MOG2 to isolate moving objects (vehicles). The steps include:

- **Grayscale Conversion and Gaussian Blur:** Each frame is converted to grayscale to simplify processing, followed by Gaussian blur to reduce noise and smooth the image.
- **Background Subtraction:** A background subtraction algorithm is applied to detect moving objects by subtracting the current frame from a dynamically maintained background model.
- **Contour Detection:** Contours are drawn around the detected foreground objects, which typically correspond to vehicles in motion.

- **Contour Filtering by Area:** Small contours likely caused by noise, shadows, or minor movement are removed using an area threshold to ensure only significant vehicles are counted.

3.1.2 Approach 2: Optical Flow-Based Motion Estimation

This method uses dense optical flow to estimate vehicle movement between consecutive video frames. By analyzing how pixel values shift across frames, we can determine the direction and magnitude of motion, which helps estimate traffic flow. The Farneback algorithm is employed for its efficiency and ability to capture subtle movements. The overall density of traffic is inferred from the average magnitude of motion across the frame higher values typically indicate heavier or faster-moving traffic.

- **Grayscale Conversion:** Frames are converted to grayscale to simplify optical flow calculations.
- **Optical Flow Computation:** The Farneback method computes dense motion vectors between pairs of frames.
- **Motion Vector Decomposition:** Motion vectors are split into magnitude and direction components.
- **Visualization:** Motion is visualized using HSV mapping, where hue represents direction and brightness indicates magnitude.
- **Density Estimation:** The average magnitude of motion vectors is used to classify traffic density.

3.2 Deep Learning-Based Approaches

In addition to traditional computer vision techniques, we implemented multiple Deep Learning (DL) models known for their effectiveness in object detection. These models leverage convolutional neural networks (CNNs) and transformer architectures to automatically learn features and accurately detect vehicles within video frames. Each model was pre-trained on large-scale datasets and fine-tuned or evaluated on traffic footage for vehicle detection.

- **YOLOv8 (You Only Look Once - version 8):** YOLOv8 is an advanced real-time object detector that provides a strong balance between detection accuracy and processing speed. It predicts bounding boxes and class probabilities in a single pass through the network, making it ideal for real-time applications such as traffic monitoring.
- **RetinaNet (ResNet-50 backbone):** RetinaNet addresses the problem of class imbalance using a unique focal loss function. With a ResNet-50 backbone, it achieves high detection accuracy even in scenes with varying vehicle sizes and densities.
- **DETR (DEtection TRansformer):** DETR uses a transformer-based architecture that integrates convolutional features with self-attention mechanisms. DETR predicts objects directly, making it more robust to overlapping vehicles and occlusions.
- **SSD (Single Shot Detector):** SSD performs object detection in a single forward pass, which allows for faster inference times. It discretizes output space into a set of default boxes and predicts both class scores and box offsets, making it well-suited for low-latency traffic analysis tasks.
- **EfficientDetr:** EfficientDetr is part of a family of object detectors optimized for speed and model size through compound scaling. It is particularly suitable for edge-device deployment, maintaining competitive accuracy while keeping computational cost low.
- **Faster R-CNN:** A two-stage detector where the first stage proposes regions of interest and the second stage classifies and refines these proposals. Though slower than single-shot detectors, it offers high accuracy and is especially effective in densely packed traffic scenes.
- **R-CNN (Region-based Convolutional Neural Network):** One of the earliest region proposal-based object detectors. It extracts region proposals and passes each through a CNN for classification and bounding box regression. Although it set the foundation for modern detectors, it is computationally intensive and not suited for real-time traffic monitoring.

All models were evaluated on the same video input, allowing us to compare their ability to detect vehicles under different lighting, density, and movement conditions. The detected vehicle count in each frame was then mapped to a traffic density category Low, Medium, or High mirroring the classification used in the CV approaches but with improved precision and generalization.

Note: While most DL models performed robustly, the original **R-CNN** model did not yield suitable results. Its region proposal and classification process is significantly slower and less optimized compared to modern detectors. This led to missed detections, particularly in real-time sequences, making it impractical for our use case. Additionally, the model struggled with overlapping objects and small-scale vehicles due to its older architecture and lack of feature pyramids or advanced post-processing techniques.

4 Results

Traffic density estimation was successfully implemented using both Computer Vision (CV) and Deep Learning (DL) based approaches.

For the CV-based pipeline, two primary techniques were adopted: Background Subtraction with Contour Detection and Optical Flow-Based Motion Estimation. Both methods were able to process traffic footage and classify density levels based on the observed motion or vehicle count in each frame.

The DL-based approach utilized several state-of-the-art object detection models including YOLOv8, RetinaNet, DETR, SSD, EfficientDet, and Faster R-CNN. These models were capable of accurately detecting vehicles in each frame, enabling effective traffic density estimation through vehicle counting.

The combination of these approaches demonstrated the adaptability of the system to different conditions and input types, fulfilling the project objective of reliable and real-time traffic analysis.

5 Deployment

The complete system was deployed using **Streamlit**, a Python-based web application framework, which enabled an interactive and user-friendly interface. This platform allowed for seamless integration of both Computer Vision and Deep Learning pipelines in a web-based environment. Key functionalities of the deployed application include:

- Uploading traffic video footage directly through the interface for analysis.
- Real-time visualization of the vehicle count, traffic density level (Low, Medium, High), and live frame analysis.

6 Project Links

- **GitHub Repository:** `https://github.com/Prajjwal-dixit/traffic-density-estimation`
- **Deployment Link (CV Approach 1):** `https://cvproject-v4k6n5lfk8eyddfg4q6appk.streamlit.app/`
- **Deployment Link (CV Approach 2):** `https://cvproject-6kwpgl7oznihbmwgsxnglv.streamlit.app/`
- **YOLOv8:** `https://drive.google.com/file/d/1whwhA2j1NNiWdWQzKcLO5q_ijpmAyvia/view?usp=drive_link`
- **Detr:** `https://drive.google.com/file/d/1ajIgO9xBMmxGsWhmdNpqM_cSgxOMAt9S/view?usp=drive_link`
- **Faster R-CNN:** `https://drive.google.com/file/d/1tnJl5-TJgRRZMueTx3nYpMOICOAFxjFs/view?usp=drive_link`
- **RetinaNet:** `https://drive.google.com/file/d/16yKOwDurRBLlA5tBzMfaORNRWeBRh3Gh/view`
- **SSD:** `https://drive.google.com/file/d/1Llv5A3h3iD19Zcmo8TrQ2e6xI7p1NkOZ/view`
- **EfficientDetr:** `https://drive.google.com/file/d/1VLoabWifRXn7GDqGwUOa8HBmF85I6XwS/view`

Note: Deployment of deep learning models on Streamlit was avoided as they consistently caused the application to crash.

7 Screenshots from the Deployed Website

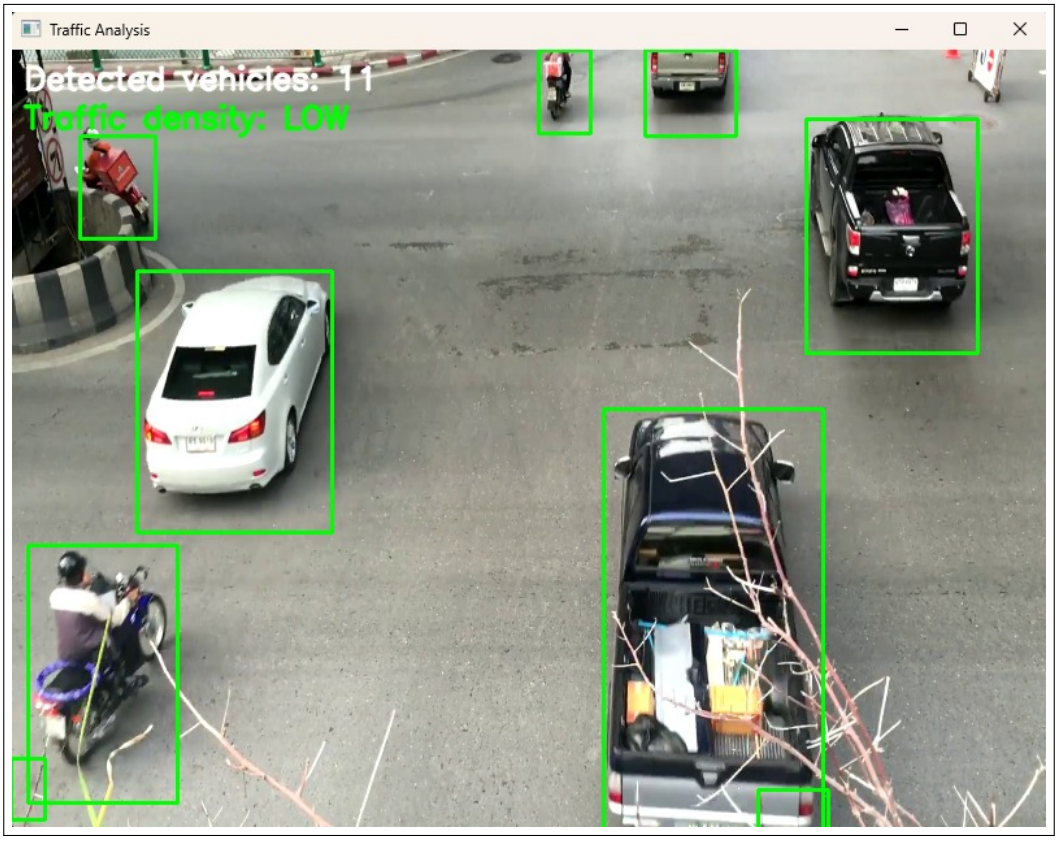

Figure 1: CV approach 1 (Background Subtraction and Contour Detection)

As observed during testing, the model occasionally detects not only entire vehicles but also individual parts (such as tires or shadows) as separate vehicles. This leads to duplicate detections and, consequently, a slight reduction in the overall accuracy of the traffic density estimation.

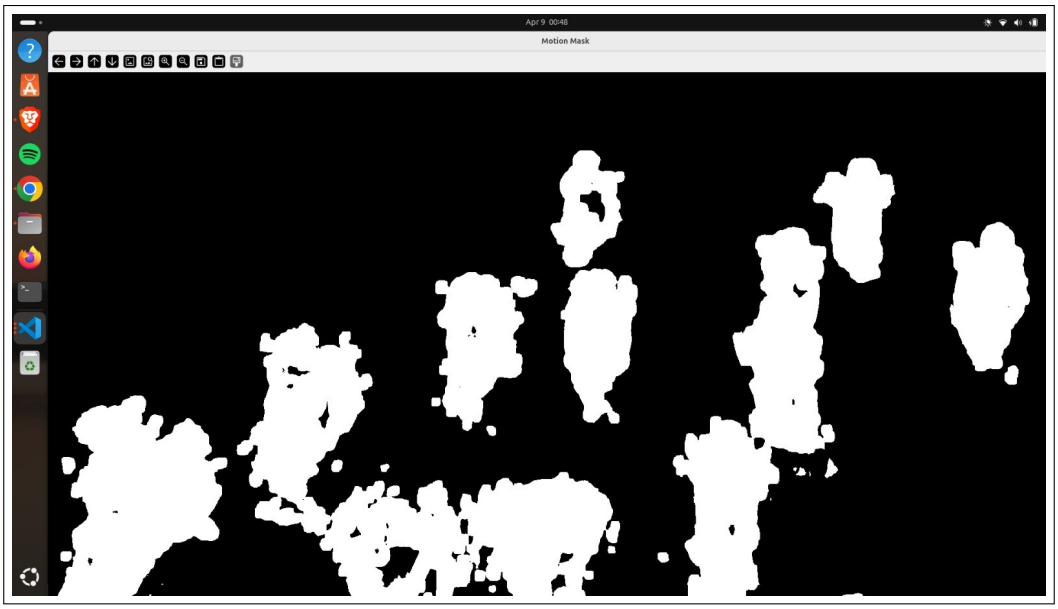

Figure 2: CV approach 2 (Optical Flow-Based Motion Estimation) Motion Mask

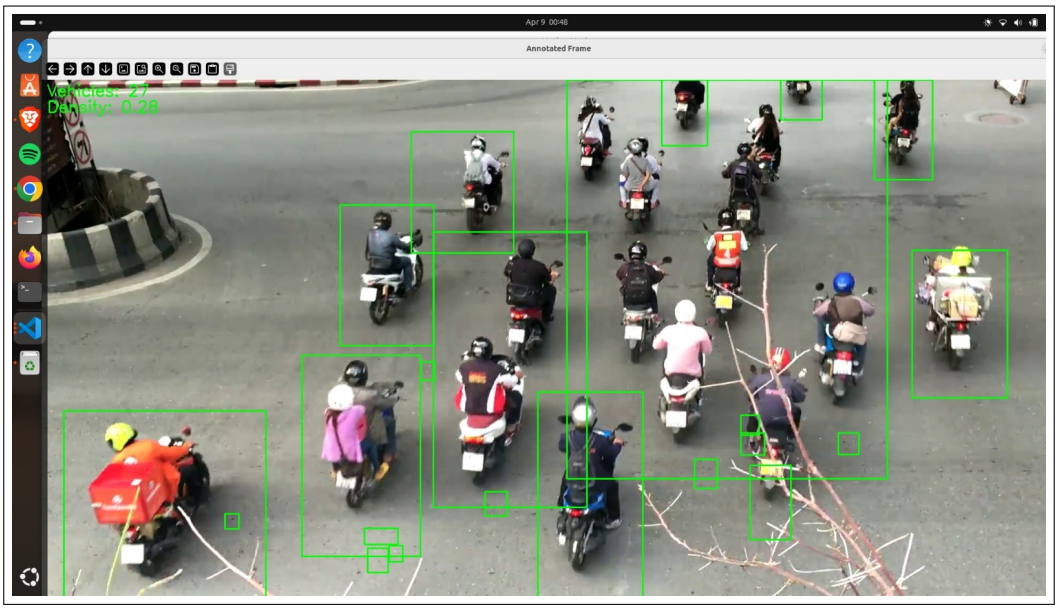

Figure 3: CV approach 2 (Optical Flow-Based Motion Estimation) Output

Figure 4: Output of R-CNN Model

R-CNN model did not yield suitable results. Its region proposal and classification process is significantly slower and less optimized compared to modern detectors. This led to missed detections, particularly in real-time sequences, making it impractical for our use case. Additionally, the model struggled with overlapping objects and small-scale vehicles due to its older architecture and lack of feature pyramids or advanced post-processing techniques.

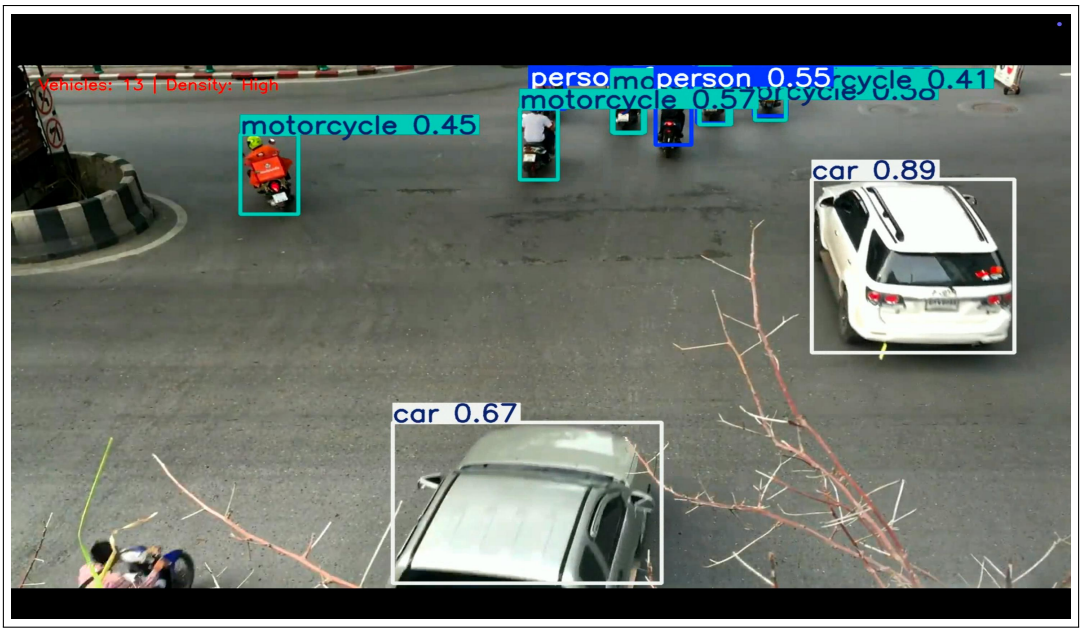

Figure 5: Output of YOLOv8 Model

Figure 6: Output of Detr Model

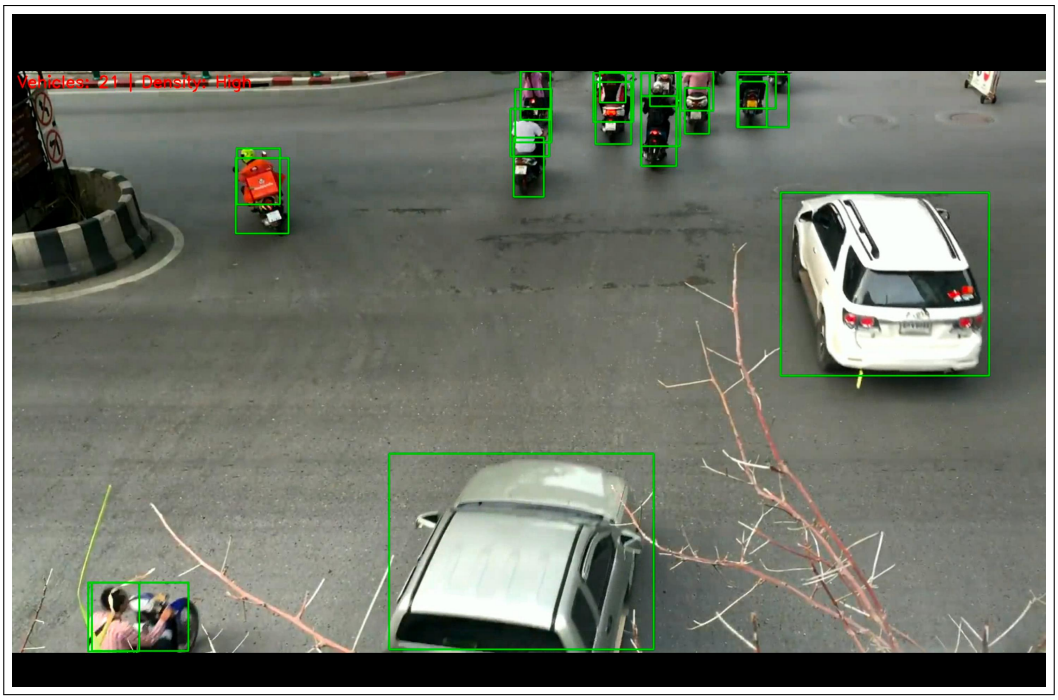

Figure 7: Output of Faster R-CNN Model

Figure 8: Output of Retina Net Model

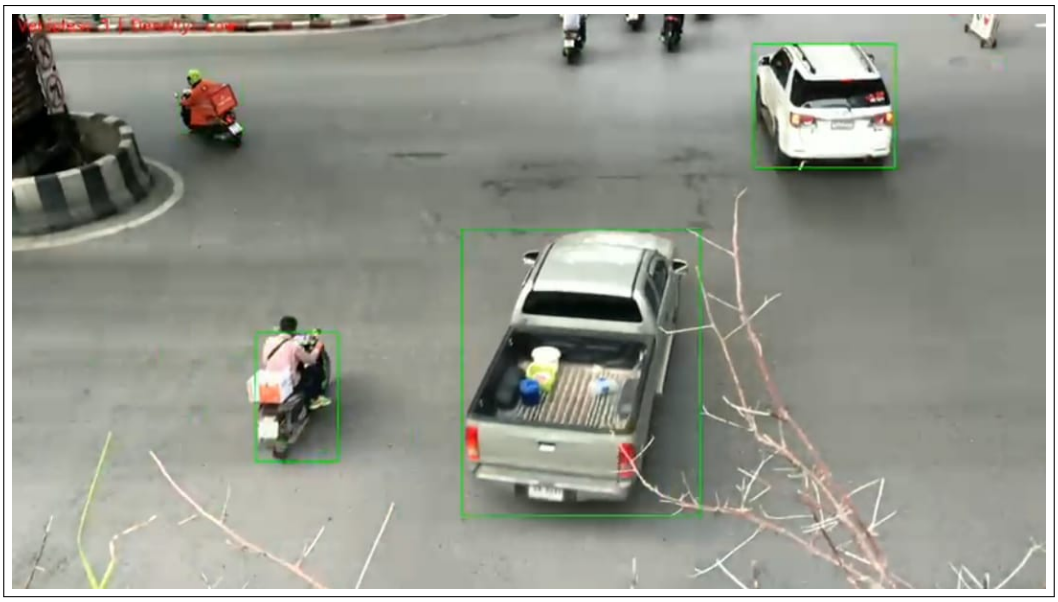

Figure 9: Output of SSD Model

Figure 10: Output of Efficient Detr Model

8 Team Members and Contributions

- **Vasishtha Pandya:**
  Developed RetinaNet and Single Shot Detector (SSD) models for vehicle detection, contributed to refining model parameters, and contributed in research and evaluation of various Deep Learning-based approaches for traffic density estimation. Also involved in report writing and documentation.

- **Prajjwal Dixit:**
  Implemented traffic analysis using background subtraction and contour detection techniques to identify and track vehicles in video streams. Developed and deployed an interactive web application using Streamlit, enabling real-time visualization and ease of access through a user-friendly interface.

- **Keshav Khandelwal:**
  Implemented the optical flow-based motion detection system using the Farneback method in OpenCV. Additionally, deployed the solution using Streamlit to make it interactive and easily accessible through a web interface. Also contibuted in reasearch of Computer Vision based approaches.

- **Raj Vijayvargiya:**
  Developed and fine-tuned YOLOv8 and Faster R-CNN models for vehicle detection, and contributed significantly to the research, evaluation, and comparative analysis of Deep Learning-based approaches for accurate traffic density estimation. Also involved in implementation of optical flow based motion detection system in OpenCV.

- **Raghavi Epuri:**
  Designed and optimized DETR and EfficientDetr models for accurate vehicle detection, focusing on performance tuning and model evaluation and contributed to the research and analysis of Deep Learning-based techniques for robust traffic density estimation. Also involved in report writing and documentation.

9 Conclusion

This project demonstrates the practical value of integrating traditional computer vision (CV) techniques with advanced deep learning (DL) models to estimate traffic density accurately. The CV-based approaches offer lightweight and fast processing, making them suitable for environments with limited computational resources. They work well in scenarios where the background is relatively static and lighting conditions are stable. On the other hand, deep learning models especially those like YOLOv8, EfficientDet, and Faster R-CNN excel in complex environments, handling occlusions, dynamic lighting, and dense traffic scenes with greater precision. By combining the strengths of both approaches, the system ensures flexibility, adaptability, and improved reliability in varying real-world conditions.

The deployment of the system using Streamlit adds a user-friendly and scalable interface that allows users to test and visualize results without complex setup. This makes the tool accessible for traffic analysts, urban planners, or researchers aiming to assess traffic congestion. Future improvements could include integrating live traffic feeds from IP cameras or drones, optimizing model performance for edge devices, and enhancing robustness in challenging conditions such as nighttime traffic, heavy rain, or shadows. Additionally, incorporating temporal information across frames using object tracking could lead to even more accurate and consistent traffic flow analysis.