SUMOFlow AI – Initial System Design & Work Division Report

Team: 46

Project Title: SUMOFlow AI: AI-Driven Traffic Optimization for Next-Generation Smart Cities: A YOLO-SUMO Integration Approach

Team Members:

- Rana waleed 202201737
- Roaa Raafat 202202079
- Mariam Alhaj 202200529

Supervisor: Dr. Mohamed Maher

GitHub Repository: https://github.com/ranna-waleed/sumoflow-ai-traffic-optimization

Table of Contents

1. Project Summary

1.1 Objective: SUMOFlow AI is a project that aims to design and prototype a traffic optimization system, which uses AI in real-time to optimize traffic signal timing to decrease urban congestion, fuel consumption and CO2 emissions. The system uses computer vision to detect vehicles and simulate traffic in order to intelligently optimize the pattern of signals.

1.2 Scope: The project has three main areas: (1) building a real-time vehicle detection and counting module with computer vision models; in which the system would be tested to assess three different detection models: YOLO, Fast R-CNN, and Faster R-CNN (2) integration with the SUMO traffic simulation platform to simulate urban conditions and test adaptive optimization strategies (3) implementation of intelligent traffic signal control algorithms to respond dynamically to real-time traffic conditions. This prototype will prove that it is feasible via simulated scenarios checking the reduction that will be made in the average wait time, queue length, and reduction in emissions as compared to the traditional fixed timing signal systems.

2. Progress Since Proposal

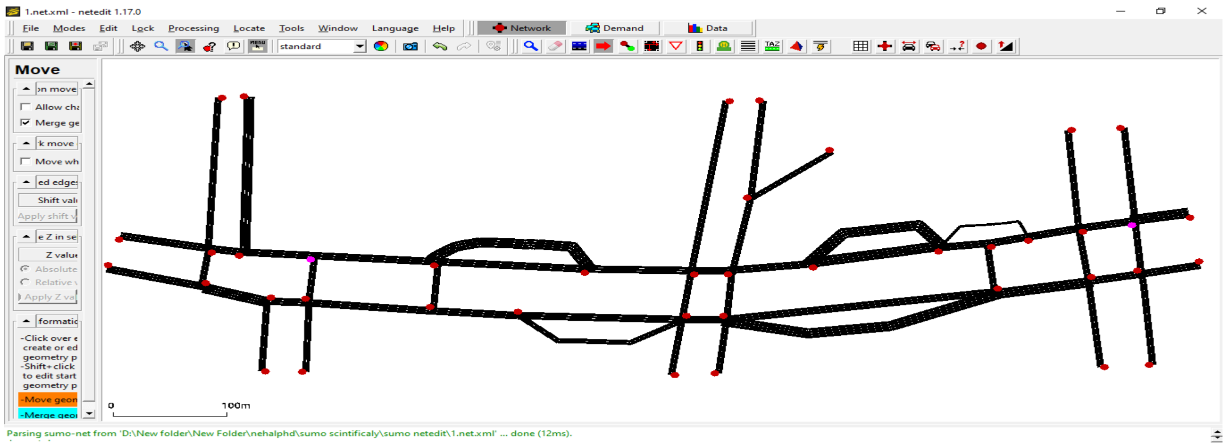
2.1 What Has Been Completed

Structured SUMO Learning Sessions: Our team completed multiple collaborative online learning sessions covering SUMO fundamentals. We were also introduced to network file format (.net.xml), created road networks with netedit, learned about route file formats (.rou.xml), learned about traffic light, had a hand-on experience with TraCI Python and simulation control, and learned the SUMO output file format metrics.

System Architecture and Design Work: System architecture diagrams were created that identified all the major components and interactions.

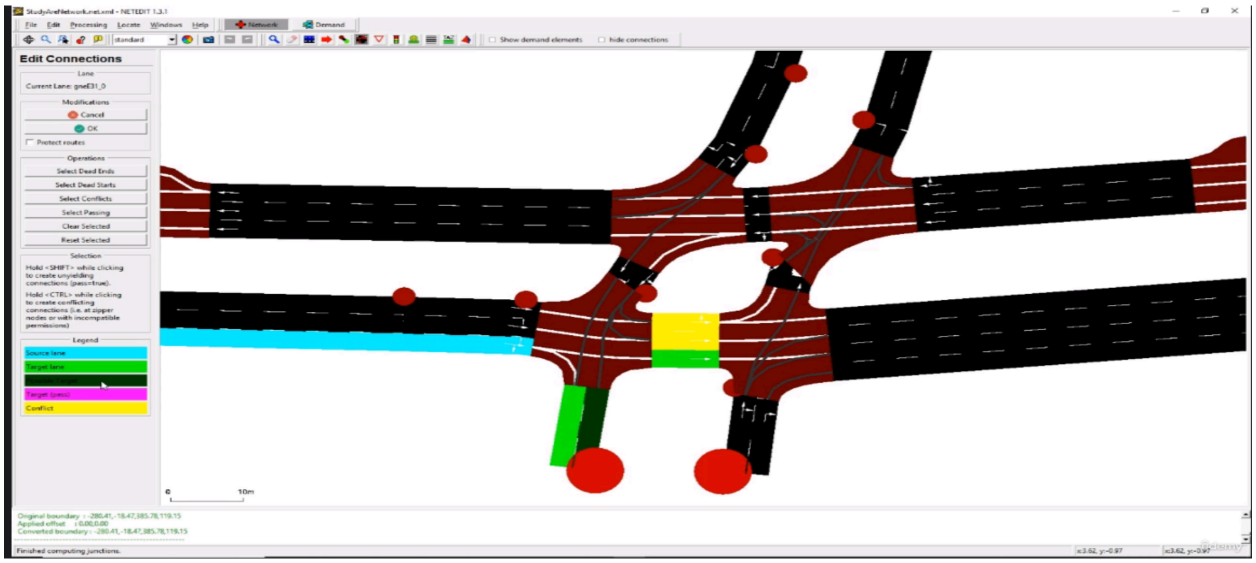
2.2 Evidence of Progress

Our team's progress is evidenced by some key deliverables from our SUMO learning phase. We successfully designed practice road networks (Fig. 1), configured 4-way intersections (Fig. 2), and validated our setup through live simulations (Fig. 3).


1.net.xml - netedit 1.17.0
File Modes Edit Lock Processing Locate Tools Window Language Help
Network Demand Data
standard
Move
0 100m

```
Parsing sumo-net from 'D:\New folder\New Folder\nehalphd\sumo scintificaly\sumo netedit\1.net.xml' ... done (12ms).
```

Fig. 1 : SUMO netedit interface showing team's practice road network design with multiple intersections and lane configurations



```
Original boundary : -280.41,-18.47,385.78,119.15
Applied offset    : 0.00,0.00
Converted boundary : -280.41,-18.47,385.78,119.15
```

```
Finished computing junctions.
```

Fig. 2: SUMO netedit interface showing the configured 4-way intersection.

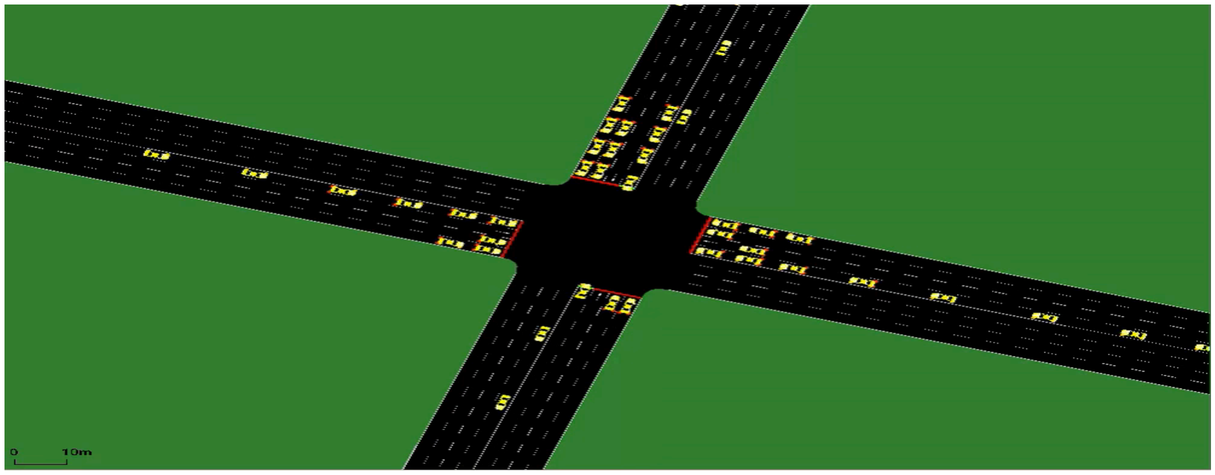


Fig. 3: Live SUMO simulation showing vehicle distribution and movement patterns at the configured intersection.

3. System Architecture

The architecture consists of five major layers organized in a hierarchical structure that processes data from raw video input through optimization to traffic signal control:

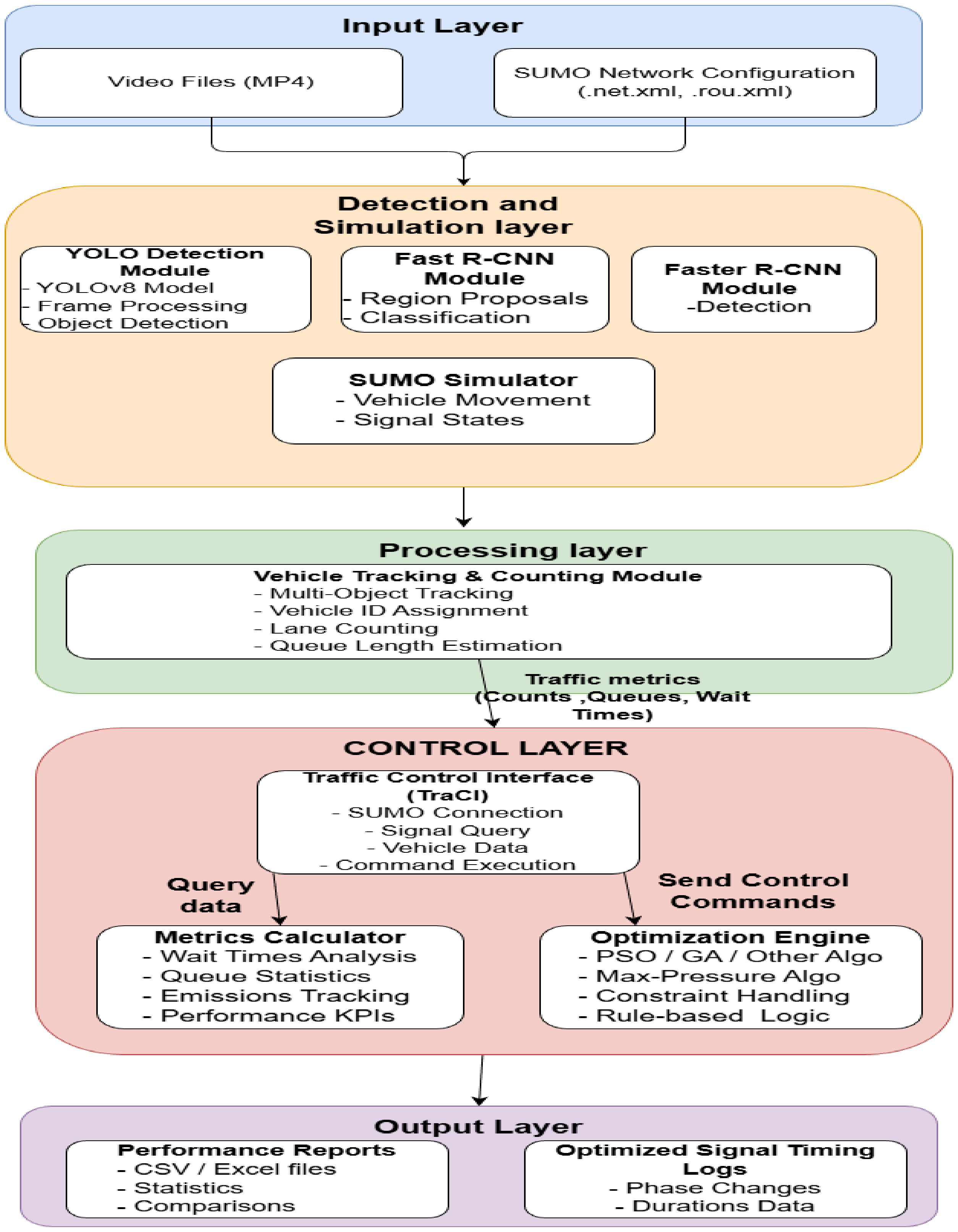


Fig. 4: System Architecture Diagram

4. Component Breakdown

4.1 Detection Modules (YOLO, R-CNN, Fast R-CNN, Faster R-CNN)

Function: Real-time vehicle detection and classification from video feeds.

Inputs: Video streams , frame resolution , frame rate , region of interest coordinates.

Expected Output: Vehicle bounding boxes, class labels (car, truck, bus, motorcycle), timestamps, detection counts per frame in JSON/CSV format.

Main Functionality: Load pre-trained YOLO weights, process frames at intervals, apply non-maximum suppression, export structured results with timestamps.

Technologies: YOLOv8, Fast R-CNN, Faster R-CNN, PyTorch, OpenCV.

Justification: We selected this suite of models to compare the inference speed of YOLO against the detection accuracy of the R-CNN family. This ensures we identify the optimal balance required for real-time traffic signal control.

4.2 Vehicle Tracking & Counting Module

Function: Track vehicles across frames, assign unique IDs, count vehicles per lane, estimate queue lengths.

Inputs: YOLO detections (bounding boxes, labels, confidence), frame timestamps, lane configurations.

Expected Output: Unique vehicle tracks with lane-specific counts, queue lengths, lane occupancy rates, average speeds.

Main Functionality: Implement tracking, assign unique IDs, count line-crossing events, identify stationary vehicles for queue estimation, calculate occupancy rates.

Technologies: NumPy, SciPy, OpenCV, Pandas.

4.3 SUMO Simulator

Function: Simulate realistic urban traffic with configurable networks and vehicle flows.

Inputs: Network file (.net.xml), route file (.rou.xml), traffic light (.add.xml), simulation config (.sumocfg), vehicle behavior parameters.

Expected Output: Vehicle positions and states, traffic light states, traffic metrics (wait time, speed, distance), emissions data (CO_2, NO), and summary statistics.

Main Functionality: Load network, generate vehicle flows, execute time-step simulation with car-following models, handle intersection logic, support external control via TraCI data.

Technologies: SUMO , TraCI Python API, netedit, XML parsing libraries.

Justification: SUMO was chosen because it is open-source and provides the TraCI (Traffic Control Interface) API. Unlike other simulators, TraCI allows our Python algorithms to override traffic lights frame-by-frame, which is strictly necessary for testing dynamic AI-based optimization.

4.4 Traffic Control Interface (TraCI)

Function: Bidirectional communication between SUMO and optimization logic.

Inputs: Running SUMO instance, connection parameters , vehicle count data, optimization commands.

Expected Output: Current traffic light states with phase and duration, vehicle data (IDs, positions, speeds), command confirmations, simulation time.

Main Functionality: provide functions for vehicle counts and signal states, send control commands to change phases and durations.

Technologies: TraCI Python library, JSON.

4.5 Optimization Algorithms

Function: Analyze traffic conditions and compute optimal signal timing decisions.

Inputs: Vehicle counts per lane, queue lengths, wait times, system constraints (min/max green time, yellow intervals).

Expected Output: Optimized phase sequence, green time durations, cycle length, priority assignments.

Main Functionality: Implement optimizer algorithms, use rule-based logic for green extension and early termination, generate and send control commands via TraCI.

Technologies: Python , NumPy, Pandas, SciPy, algorithm implementations.

4.6 Metrics Calculator

Function: Compute performance metrics and generate comparison reports.

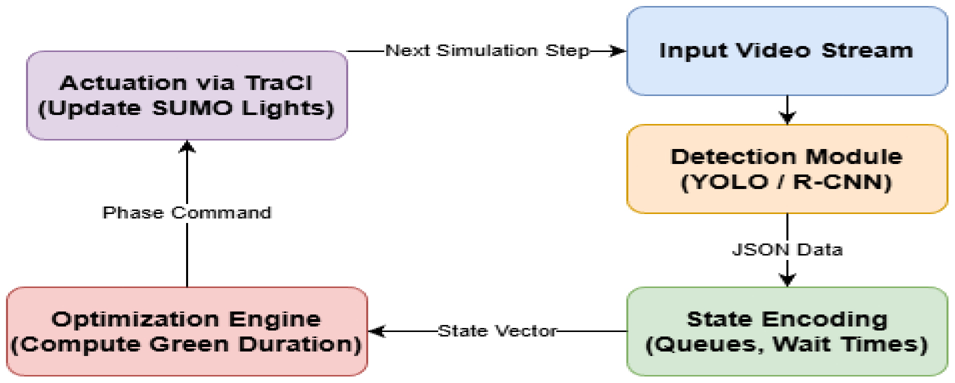
Inputs: signal timing logs, SUMO tripinfo output, emissions output, queue measurements, baseline configuration.

Expected Output: wait time distributions, queue statistics, travel times, emissions totals (CO_2, fuel), percentage improvements vs baseline, statistical significance tests, time-series data.

Main Functionality: Parse SUMO XML outputs (tripinfo, emissions), calculate statistical (mean, median, percentiles), process queue time-series, aggregate emissions, run baseline comparisons, perform significance tests, generate CSV/Excel reports with visualizations.

Technologies: Pandas, NumPy, Matplotlib/Seaborn, SciPy.

5. Data Flow



Actuation via TraCI (Update SUMO Lights)
Next Simulation Step
Input Video Stream
Detection Module (YOLO / R-CNN)
Phase Command
JSON Data
Optimization Engine (Compute Green Duration)
State Vector
State Encoding (Queues, Wait Times)

Fig. 5: Data Flow Diagram showing the closed-loop feedback system between video input, detection module, state encoding, optimization engine, and SUMO traffic light actuation via TraCI.

1. Multi-Model Perception: Raw video is processed by the selected detection model (YOLO, R-CNN, Fast R-CNN, or Faster R-CNN) to extract vehicle coordinates.
2. State Encoding: Detections are aggregated into a 'State Vector' (queue lengths, waiting times).
3. Optimization: The algorithm processes this vector to compute the optimal Green Phase duration.
4. Actuation (TraCI): The decision is sent to SUMO, instantly updating the traffic light state for the next simulation step.

6. Work Breakdown Structure & Equity

TABLE I Work Breakdown Structure and Task Allocation

Task	Task Description	Responsible	Start Date	End Date	Status
Setup	1. Install Python & SUMO 2. Set environment (OpenCV, PyTorch)	All Team Members will share these tasks	10 Oct	25 Oct	Completed
SUMO Network	1. Build 4-way intersection 2. Configure traffic lights 3. Create routes 4. Test basic SUMO simulation	All Team Members will share these tasks	3 Dec	30 Jan	In Progress
YOLO Detection	1. Implement YOLO pipeline 2. Frame extraction 3. Run inference 4. Export results to JSON 5. Basic accuracy tuning	Rana	31 Jan	15 Feb	In Progress
Fast R-CNN Detection	1. Prepare dataset format 2. Implement Fast R-CNN inference Process bounding boxes 3. Integrate outputs with pipeline	Roaa	31 Jan	15 Feb	In Progress
Faster R-CNN Detection	1. Load pre-trained Faster R-CNN 2. Run inference	Mariam	31 Jan	15 Feb	In Progress

Tracking	1. Vehicle counting 2. Queue estimation 3. Integrate with all detectors	Rana	16 Feb	29 Feb	Not Started
TraCI Integration	1. SUMO–Python connection 2. Query signals/vehicles 3. Sync simulation steps	Roaa	1 Mar	20 Mar	Not Started
Optimization Engine	1. Implement Optimizer like PSO / GA 2. Connect optimization to TraCI	Mariam	21 Mar	31 Mar	Not Started
Metrics & Analysis	1. Parse SUMO outputs 2. Compute KPIs (wait time, queues, emissions) 3. Compare detectors & algorithms	Rana / Roaa	1 April	15 Apr	Not Started
Integration & Testing	1. Integrate all modules 2. Perform end-to-end tests 3. Debug 4. Performance optimization	All Team Members will share these tasks	16 Apr	30 Apr	Not Started
Validation & Finalization	1. Run final experiments 2. Create visualizations	Mariam	1 May	31 May	Not Started

The work distribution in Table I ensures equity by assigning each member leadership of a distinct detection model while sharing critical phases collaboratively. Rana leads YOLO and tracking, Roaa handles Fast R-CNN and TraCI, and Mariam manages Faster R-CNN and optimization. This parallel structure enables efficient development while shared tasks ensure collective knowledge.

7. Risk Analysis & Mitigation

7.1 YOLO Detection Accuracy Problems

Category: Technical

Description: Pre-trained YOLO can be challenged by changing lighting, unusual vehicles, and weather conditions, leading to incorrect counts which will lead to poor optimization.

Mitigation: tune confidence thresholds, implement temporal filtering across frames,, design optimization to handle imperfect data gracefully.

7.2 Insufficient Computational Resources

Category: Resource

Description: YOLO and SUMO together may exceed available computing power, causing slow processing, simulation lag, or crashes.

Mitigation: Use lightweight models , reduce frame rate , leverage free cloud resources (Colab, Kaggle), optimize code efficiency.

7.3 Team Skill Gaps

Category: Resource

Description: Lack of experience with YOLO, TraCI, tracking algorithms, and optimization may slow progress and cause implementation mistakes.

Mitigation: Continue structured learning sessions, practice with simplified examples first, utilize online resources.

7.4 Integration & Timeline Delays

Category: Timeline

Description: Training and integrating four distinct detection models (R-CNN through YOLO) plus the optimization engine may exceed the semester timeline.

Mitigation: We have prioritized the **YOLO** integration first as the "Minimum Viable Product" (MVP). If delays occur, we will finalize the system with YOLO and benchmark the R-CNN family offline, rather than blocking the main simulation pipeline.

References

[1] O. A. Ayegbusi et al., "A Deep Learning-based Model for Traffic Signal Control using the YOLO Algorithm," International Journal of Computer Applications, vol. 186, no. 63, pp. 43-54, Jan. 2025, doi: 10.5120/ijca2025924452.

[2] M. T. Jahangir, Tahreem Fatima, and Qandeel Fatima, “Evaluating Faster R-CNN and YOLOv8 for Traffic Object Detection and Class-Based Counting”, IJIST, vol. 6, no. 4, pp. 1606–1620, Oct. 2024.