International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 06 | Jun 2026 www.irjet.net p-ISSN: 2395-0072

Density Based Smart Traffic Management System

Prof. Santosh Dattatray Kale[1], Atharva Madhukar Deokate[2], Pratiksha Dattatray Wanave[3], Harshada Ravindra Wadkar[4]

[1]Professor, [234]Department of Electronics & Telecommunication Engineering SVPM's College of Engineering, Malegaon (Bk), Savitribai Phule Pune University, Pune, India

---

Abstract: *Urban traffic congestion has emerged as one of the most pressing infrastructural challenges of the twenty-first century, driven by rapid population growth and an ever-increasing vehicle population that far outpaces the expansion of road networks. Conventional fixed-time signal systems, which allocate identical green-light durations to every lane regardless of prevailing traffic conditions, are fundamentally ill-suited to the dynamic, non-uniform nature of real-world traffic demand. This paper describes the design, implementation, and evaluation of an AI-powered, density-aware smart traffic management system that uses live vehicle detection through computer vision and deep learning to overcome these shortcomings. A Raspberry Pi 4 serves as the edge-computing core, acquiring continuous video from a wide-angle camera module and running a YOLOv4-tiny object-detection pipeline to estimate per-lane vehicle density in real time. Derived density indices are relayed over a UART link to an ESP32 microcontroller, which dynamically scales green-light durations - lengthening them for congested lanes and curtailing them for lightly loaded ones. A servo-motor-controlled barrier at the stop line and a 16x2 I2C LCD panel round out the physical interface, offering both automated access control and driver-facing status information. Simulation-based evaluation against a conventional fixed-time baseline confirms substantial reductions in average waiting time, measurable gains in intersection throughput, and lower per-vehicle fuel consumption. The resulting architecture is compact, low-cost, and readily extensible within broader IoT-enabled smart-city deployments.*

Keywords: Smart Traffic Management, Vehicle Density Detection, YOLO Object Detection, Raspberry Pi, ESP32, IoT, Adaptive Traffic Signals, Computer Vision, Deep Learning, Congestion Control

I. INTRODUCTION

The steady expansion of urban centres worldwide has brought with it a dramatic surge in motorised traffic, placing mounting strain on road infrastructure that was rarely designed with today's vehicle densities in mind. As the World Health Organization has documented, traffic congestion in cities generates substantial economic losses through wasted commuter time and excess fuel burn, while simultaneously elevating ambient pollution levels in ways that carry measurable public-health consequences [1]. At the heart of this problem lies a mismatch between the rigid, pre-programmed nature of conventional traffic signals and the highly variable patterns of vehicle arrival at intersections. Fixed-cycle controllers allocate the same green-light window to every lane on every cycle, with no awareness of whether a particular lane is backed up for hundreds of metres or is entirely empty. The result is a well-documented and widespread inefficiency: congested lanes receive no extra clearance time, while lightly trafficked lanes consume green time that yields little benefit.

Adaptive signal control, underpinned by computer vision and machine learning, offers a principled route out of this impasse. By continuously measuring the number of vehicles queued in each approach lane and feeding that information back into the timing algorithm, an adaptive controller can redistribute green time toward where it is most needed, cycle by cycle [2]. The practical feasibility of this approach has been transformed in recent years by the emergence of accurate, low-latency deep-learning detectors - most notably the You Only Look Once (YOLO) family - that can run on affordable, low-power embedded hardware with throughput sufficient for intersection-level control [3]. What was once a research curiosity requiring dedicated GPU servers can now be realised with a single-board computer costing a fraction of the price.

Building on these advances, this paper presents an end-to-end hardware-software system for density-based adaptive traffic management. A Raspberry Pi 4 handles vision processing and density estimation; an ESP32 microcontroller translates the resulting timing decisions into physical signal actuation; and a servo-motor barrier together with an LCD display extend the system's reach beyond simple LED signals. The remainder of the paper is structured as follows. Section II surveys related work across the principal technical threads feeding into this design. Section III articulates the problem that motivates the work. Section IV presents the proposed system architecture. Section V states the design objectives. Section VI describes the methodology adopted during development and evaluation. Section VII covers the implementation in detail. Section VIII reports

simulation results and discusses their implications. Sections IX and X provide the conclusion and future scope respectively, followed by the full reference list.

II. LITERATURE REVIEW

Intelligent traffic signal control has attracted sustained research interest for several decades, with the field evolving through three broad technological generations: sensor-based counting, image-processing-based estimation, and deep-learning-driven detection. A selective review of representative work from each thread, together with emerging IoT-integrated approaches, frames the contribution of the present paper.

A. Sensor-Based Adaptive Systems

The earliest adaptive controllers relied on inductive loop detectors buried beneath the road surface or, later, on infrared and ultrasonic proximity sensors mounted above it. Loop detectors offered reliable axle-counting over decades of practical deployment, yet their installation demanded costly and disruptive roadworks, and their spatial coverage was inherently limited to the point at which the loop was embedded [4]. Ultrasonic alternatives somewhat eased the deployment burden and could provide crude queue-length estimates, but they proved sensitive to environmental noise and were effectively restricted to measuring a single lane at a time [5]. Both technologies therefore share a fundamental scalability constraint: instrumenting a multi-lane intersection requires multiplying the sensor count, with a corresponding multiplication of installation and maintenance cost.

B. Image-Processing-Based Approaches

The widespread availability of low-cost cameras and capable embedded processors opened the door to vision-based traffic analysis that requires no intrusive road modification. Jain and Verma demonstrated a vehicle-density controller built around grayscale background-subtraction applied to intersection frames, reporting an approximately 23% reduction in average waiting time relative to a fixed-time baseline - a promising result, though one that degraded noticeably under poor lighting or when vehicles heavily overlapped in the image plane [6]. Patil et al. extended this line of work with a frame-differencing and blob-counting scheme that retained adequate accuracy on ARM-class processors while keeping computational demand low enough for real-time operation [7]. These approaches established the viability of camera-based density estimation but left open the question of how to sustain performance across the wide range of illumination and occlusion conditions encountered in practice.

C. Deep Learning and CNN-Based Detection

The introduction of the YOLO detection framework by Redmon et al. marked a watershed moment for vision-based traffic analysis [8]. By framing detection as a single regression pass over the full image rather than a sequential region-proposal pipeline, YOLO achieved real-time throughput exceeding 45 frames per second on contemporary GPU hardware without sacrificing competitive accuracy. Subsequent iterations refined the architecture in various ways; notably, the YOLOv4 variant of Bochkovskiy et al. incorporated a suite of training and architectural enhancements that pushed the accuracy-speed trade-off further in both directions, making it particularly attractive for deployment on resource-constrained edge devices [10]. Li and Wang applied a CNN-based signal timing model to urban intersections and demonstrated that lane-wise vehicle counting paired with adaptive duration assignment yielded measurable improvements in traffic flow efficiency, validating the translation of detection capability into operational benefit [9].

D. LSTM and Predictive Traffic Control

A limitation shared by all purely reactive controllers - including those based on YOLO detection - is that they respond to congestion only after it has formed. Patel and Joshi addressed this limitation by training a Long Short-Term Memory network on historical and real-time traffic streams to anticipate congestion before it reaches critical levels, enabling preemptive extension of green phases during predicted peak periods [11]. The approach introduced a genuinely predictive element into signal control, reducing peak-hour delays beyond what reactive methods could achieve. While the present paper does not incorporate temporal prediction, the results of Patel and Joshi motivate its inclusion as a future enhancement.

E. IoT-Integrated and Multi-Intersection Systems

As individual intersection controllers have matured, attention has shifted toward coordinating them across road networks. Tavanaei et al. surveyed AI-based adaptive signal control with particular attention to networked deployments, noting an accelerating trend toward IoT-enabled architectures in which intersection nodes share state information to achieve system-

level optimisation [12]. Vehicle tracking across frames, rather than simple per-frame counting, has also gained traction; the Deep SORT algorithm of Wojke et al. demonstrated that associating detections over time enables reliable queue-length estimation that is more robust to momentary occlusion than frame-level methods alone [13]. Together, these contributions chart a clear trajectory from single-intersection adaptive control - the focus of the present work - toward city-scale adaptive traffic management.

III. PROBLEM STATEMENT

Despite decades of research demonstrating the limitations of fixed-time signal control, the majority of intersections in urban and peri-urban India continue to operate on pre-programmed timing plans that are reviewed and updated at most annually. The consequences are well understood by anyone who commutes regularly through a city: lanes that are heavily loaded during peak hours receive the same green allocation as they do at two in the morning, while adjacent lanes that are nearly empty consume green time that delivers minimal throughput benefit. More formally, current fixed-time systems exhibit several interrelated deficiencies that compound one another:

- Road capacity is wasted during off-peak periods and on lightly trafficked approaches, which receive more green time than they require.
- Vehicles in high-density lanes experience unnecessarily long red phases during peak hours, inflating average waiting times and promoting unsafe behaviour such as signal jumping.
- Extended idling at over-saturated intersections contributes disproportionately to vehicle exhaust emissions, with direct consequences for urban air quality.
- Emergency vehicles have no mechanism to request priority clearance, forcing ambulances and fire engines to navigate congested intersections with the same signal timing as ordinary traffic.
- Any adjustment to timing plans requires manual reprogramming by traffic engineers, introducing administrative delays and making responsive adaptation to evolving land-use patterns impractical.

A density-aware adaptive controller directly addresses each of these points. By tying green-light duration to the observed vehicle count in real time, it redistributes signal time toward where it produces the most benefit, cycle by cycle. The physical barrier mechanism and driver-facing display included in the proposed system further extend its utility beyond simple LED signal control, supporting more comprehensive intersection management.

IV. PROPOSED SYSTEM ARCHITECTURE

A. System Overview

The proposed system integrates six hardware and software subsystems into a single, hierarchically organised processing pipeline. Moving from perception to action, these are: (1) camera-based image acquisition, (2) Raspberry Pi-hosted vehicle detection and density estimation, (3) ESP32-based signal actuation, (4) RGB traffic-signal LEDs, (5) a servo-motor-controlled barrier, and (6) a 16x2 I2C LCD display. Figure 1 illustrates the overall block diagram and the data flows connecting each subsystem.

[Figure 1: Block Diagram of the Density-Based Smart Traffic Controller - Raspberry Pi to ESP32 to Traffic Signal LEDs / Servo Motor Barrier / LCD Display]

B. Camera Module

A wide-angle Raspberry Pi camera module is positioned at the head of each approach lane at a height and angle chosen to maximise the visible queue length while minimising perspective foreshortening. Operating at a configurable frame rate of 15 to 30 fps, the camera transmits raw image data to the Raspberry Pi over the Camera Serial Interface (CSI). Before passing frames to the detection model, they undergo pre-processing: resizing to the network's input resolution, channel normalisation, and conversion from BGR to RGB colour ordering.

C. Raspberry Pi (Central Processing Unit)

A Raspberry Pi 4 Model B equipped with 4 GB of LPDDR4 RAM serves as the system's computational hub. Its 1.5 GHz quad-core ARM Cortex-A72 processor comfortably sustains the YOLOv4-tiny inference workload required for lane-level density estimation without external acceleration. The processing sequence for each lane feed is as follows:

1. Frames are captured from the camera at regular polling intervals and queued for processing.
2. Each frame is submitted to the YOLOv4-tiny model, which returns bounding boxes and class labels for all detected objects within a configurable region-of-interest (ROI) mask delineating the lane boundary.
3. The raw vehicle count is normalised to a density index on a 0-100% scale, calibrated against the maximum physically realisable queue length for that lane.
4. The density index is converted to a target green-light duration via a piecewise linear mapping: indices below 30% map to the minimum green time of 15 s; indices between 30% and 70% scale linearly to durations between 15 s and 45 s; and indices above 70% map to the maximum of 60 s.
5. The computed lane index, density value, and green duration are serialised into a 4-byte packet and dispatched to the ESP32 via UART at 115,200 baud.

D. ESP32 Microcontroller (Signal Actuator)

The ESP32 occupies the actuation layer of the system, translating the timing decisions produced by the Raspberry Pi into physical signal changes. Its dual-core Xtensa LX6 architecture handles the UART receive loop on one core while the phase-state machine runs on the other, eliminating timing jitter that might otherwise arise from interrupt contention. GPIO assignments are as follows: Red LED on GPIO05, Yellow LED on GPIO18, Green LED on GPIO19, LCD SDA on GPIO02, LCD SCL on GPIO17, and servo PWM on GPIO17. To guard against communication outages - caused, for instance, by camera failure or Raspberry Pi overload - the ESP32 implements a watchdog timer with a configurable 10-second timeout; if no valid packet arrives within that window, the controller falls back to a fixed-time schedule of last-known-good parameters until communication is restored.

E. Traffic Signal LEDs

Standard RGB LEDs represent the red, yellow, and green signal phases. The green duration varies between 15 and 60 seconds depending on the density index computed for each lane, while the red and yellow durations are derived as the complement of the allocated green times to maintain a consistent total cycle length. The yellow intergreen phase, fixed at 3 seconds, provides the necessary clearance interval between conflicting green phases.

F. Servo Motor Barrier

A servo motor coupled to a 15 cm arm actuates a physical barrier at the intersection stop line. The barrier lowers to the horizontal position during red phases, providing a tangible deterrent to signal jumping, and returns to vertical during green phases to permit vehicle flow. This mechanism is particularly relevant at controlled access points such as railway crossings, toll plazas, and pedestrian-priority crossings, where a physical enforcement element meaningfully supplements the visual signal.

G. LCD Display Module

A 16x2 character LCD connected over I2C presents drivers and traffic monitors with the current lane identifier, real-time density percentage, and the remaining seconds of the active phase. The display refreshes with each new density estimate, offering a transparent window into the system's operation that supports both driver anticipation and manual oversight by traffic authorities.

V. SYSTEM OBJECTIVES

The work reported in this paper pursues the following specific objectives:

1. To design and build an autonomous traffic controller capable of deriving signal timings directly from real-time vehicle density observations, without reliance on fixed schedules or manual intervention.
2. To implement a reliable vehicle detection and density estimation pipeline using camera-based image acquisition and deep-learning inference that operates within the computational envelope of a single-board computer.
3. To reduce intersection congestion and its downstream effects - elevated waiting times, wasted fuel, and excess emissions - by allocating green time in proportion to observed lane demand, cycle by cycle.
4. To deliver the foregoing capabilities on commercially available, low-cost embedded hardware, making the system financially accessible for deployment in developing urban contexts.

5. To rigorously evaluate system performance through simulation, comparing adaptive and fixed-time control under matched traffic-demand scenarios and quantifying the improvement across relevant key performance indicators.
6. To identify the architectural extensions - predictive control, multi-intersection coordination, edge-AI optimisation - that would be most valuable in carrying the work forward toward city-scale deployment.

VI. METHODOLOGY

A. Problem Identification and Requirement Analysis

The project began with an observational study of traffic conditions at several representative urban intersections in Pune, documenting existing timing plans, measuring peak-hour and off-peak vehicle counts across all approach lanes, and recording queue lengths and waiting times during periods of known congestion. The resulting data confirmed the expected pattern: fixed-cycle signals consistently under-served high-density approaches during morning and evening peaks, leaving vehicles queued well beyond the intersection stop line while the opposing green phase cleared empty lanes. This empirical grounding informed both the performance targets set for the adaptive system and the traffic-demand parameters used to configure the simulation environment.

B. Hardware Selection and Integration

Component selection was guided by four criteria applied simultaneously: computational sufficiency for real-time inference, low idle power consumption, reasonable unit cost, and ready availability from local suppliers. The Raspberry Pi 4 satisfied the first criterion through its quad-core ARM Cortex-A72 processor, which benchmarks show to be adequate for YOLOv4-tiny at 416x416 input resolutions without a dedicated accelerator. The ESP32 addressed the actuation requirements with its abundant GPIO resources, hardware UART controllers, and an integrated Wi-Fi radio that positions the system for future cloud connectivity without additional hardware cost. Infrared proximity and ultrasonic distance sensors were retained as supplementary density-estimation channels, providing a fallback measurement path in the event of camera failure or severely degraded illumination.

C. Software Development

The image processing and detection pipeline was written in Python 3.9 using OpenCV 4.5 for frame pre-processing and the Darknet implementation of YOLOv4-tiny for object detection. The model was initialised with weights pre-trained on the MS COCO dataset, which includes dedicated classes for cars, trucks, buses, and motorcycles - the vehicle categories most relevant to Indian urban intersections. Camera-specific ROI masks were established through homographic calibration that mapped each camera's perspective projection onto a rectified top-down view of the lane, ensuring that vehicle counts reflected lane occupancy rather than apparent image density. Duplicate detections arising from partially overlapping bounding boxes were suppressed via non-maximum suppression at an IoU threshold of 0.45. To allow simultaneous processing of all lane feeds without the GIL-imposed serialization of pure CPython, each lane's detection-and-estimation loop was placed in an independent asynchronous thread, with results aggregated by a coordinating process before transmission to the ESP32.

D. Communication Protocol

Inter-board communication relies on a straightforward 115,200-baud UART link. Each packet consists of four bytes: a one-byte lane index, a one-byte density percentage (0-100), a one-byte green-time allocation in seconds (15-60), and a one-byte CRC-8 checksum computed over the preceding three bytes. The ESP32 validates every incoming packet against its checksum before acting on the contained timing data; a failed validation triggers a retransmit request rather than updating the active phase, preventing spurious timing changes from corrupting packets from propagating to the physical signals.

E. System Evaluation

Performance was assessed through microscopic traffic simulation using the SUMO (Simulation of Urban MObility) framework, which models individual vehicle dynamics and is widely accepted as a credible proxy for real intersection behaviour [14]. A four-way signalised intersection was modelled with vehicle arrival rates drawn from Poisson distributions parameterised to three traffic intensity levels: low (under 200 vehicles per hour), medium (200 to 500 vehicles per hour), and high (above 500 vehicles per hour). The distributions were calibrated against the observational data gathered in the field study phase to ensure that the simulation scenarios represented realistic conditions. Both the proposed adaptive controller and a conventional fixed-time controller with a uniform 30-second green phase were evaluated over 50 independent simulation runs per intensity level, and results were compared on three key performance indicators: mean vehicle waiting time, intersection throughput in vehicles per hour, and estimated fuel consumption per vehicle derived from published idling emission factors.

VII. IMPLEMENTATION

A. Prototype Construction

A tabletop prototype representing a four-way intersection was assembled on a 60 x 60 cm baseboard. Four Raspberry Pi camera modules were mounted at a height of 30 cm on adjustable brackets, one at each approach, with their fields of view calibrated to cover the simulated queue region for each lane. Red, yellow, and green LEDs were mounted on PVC signal frames at the intersection corners. A single servo motor with a 15 cm arm was installed at one approach to demonstrate barrier operation. The Raspberry Pi 4 and ESP32 shared a central control enclosure connected to the camera modules by CSI ribbon cables and to the signal LEDs and servo by GPIO jumper leads.

B. YOLOv4-Tiny Deployment and Calibration

YOLOv4-tiny was preferred over the full YOLOv4 model because the prototype's Raspberry Pi 4 CPU could sustain 4 to 6 inference frames per second at 416x416 resolution with the lightweight variant but fewer than 1 fps with the full model - a difference that makes the latter impractical for closed-loop control. At 4 to 6 fps, the system updates its density estimate every 160 to 250 ms per lane, a refresh interval that is comfortably fast relative to the seconds-scale phase durations of a traffic signal. Confidence and NMS thresholds were set at 0.40 and 0.45 respectively through empirical tuning on the prototype environment, balancing detection recall against the false-positive rate that tends to rise in cluttered, small-scale scenes.

C. Adaptive Signal Timing Algorithm

The timing algorithm operates on a nominal 90-second cycle, which is divided among the four approach lanes less a fixed intergreen allowance of 3 seconds per phase transition (12 seconds total across a four-phase cycle), leaving 78 seconds of productive green time per cycle. At the start of each cycle, the controller polls all four lane cameras, computes per-lane density indices, and derives green-time allocations proportionally such that each lane's share of the 78-second pool is proportional to its density index. A floor constraint prevents any lane from receiving less than the 15-second minimum, ensuring that even very lightly loaded lanes obtain a minimum service interval. The ESP32 then executes the resulting four-phase sequence, writing updated lane status, density percentage, and countdown timers to the LCD at the start of each new phase.

D. Emergency Vehicle Override

The implementation includes an optional emergency override module intended to demonstrate the extensibility of the core architecture. An electret microphone connected to the Raspberry Pi audio input captures ambient sound at the intersection; a short-time Fourier transform applied to rolling audio windows detects the characteristic 700 to 1,500 Hz siren signature used by emergency vehicles in India. On confirmed detection, the algorithm infers the likely approach lane from the audio level ratio across microphones and immediately transitions that lane to a green phase, suspending the normal adaptive cycle until the siren signature is no longer detected. This feature, while implemented in the prototype as a proof of concept, points toward the richer priority management that a production system would require.

VIII. RESULTS AND DISCUSSION

Table I summarises the mean performance metrics recorded across 50 simulation trials at each of the three traffic intensity levels, comparing the adaptive system directly with the fixed-time baseline.

Table I: Comparative Performance - Fixed-Time vs. Adaptive System

Traffic Intensity	Avg. Wait (Fixed)	Avg. Wait (Adaptive)	Reduction	Throughput Gain
Low	18.3 s	12.1 s	33.9%	+8.2%
Medium	42.6 s	24.7 s	42.0%	+17.4%
High	87.4 s	41.2 s	52.9%	+31.6%

The results tell a consistent story across all three intensity levels: the adaptive system outperforms the fixed-time baseline on every indicator, and the magnitude of the improvement grows with traffic intensity. At low volumes, where the fixed-time controller's inefficiency is least damaging, the adaptive system still reduced average waiting time by nearly 34% - a non-trivial gain attributable to eliminating wasted green time on empty lanes. At high volumes, where fixed-time control is most severely penalised by its inability to extend green phases for overloaded approaches, the improvement in average waiting time reached 52.9% and throughput increased by 31.6%. These are operationally meaningful figures: a halving of average waiting time translates directly into reduced fuel burn, lower emissions, and a perceptibly smoother commuting experience.

Fuel consumption estimates derived from published idling emission factors indicate a 38 to 47% reduction in per-vehicle fuel waste at the intersection under high-density conditions. Given that a signalised urban intersection may service tens of thousands of vehicle passes per day, even a modest per-vehicle saving aggregates into significant city-wide emission and energy benefits. The UART communication link maintained a packet error rate below 0.1% across all 8-hour endurance trials, confirming that the CRC-based integrity checking is sufficient to sustain reliable operation over extended periods. The servo barrier and LCD display operated without incident throughout the evaluation.

Two limitations were identified during prototype testing that temper these encouraging results. First, detection recall in the YOLOv4-tiny pipeline fell from approximately 94% under daylight conditions to around 76% in simulated low-light scenarios, highlighting the need for supplementary infrared illumination or a night-capable camera sensor in any production deployment. Second, in very high density conditions where vehicles queue bumper-to-bumper, the model's tendency to merge overlapping objects caused some undercounting, which in turn led to modest underestimation of the density index and correspondingly suboptimal green-time allocation. Both issues are well understood in the object detection literature and are addressed by the multi-camera fusion and enhanced lighting provisions identified in the future scope.

IX. CONCLUSION

This paper has presented a complete, functioning prototype of an AI-powered density-based smart traffic management system, designed around affordable and widely available embedded hardware. The system replaces the static, schedule-driven logic of conventional traffic controllers with a real-time feedback loop: vehicle density measured by a YOLOv4-tiny detection pipeline running on a Raspberry Pi 4 is translated into dynamically adjusted signal timings executed by an ESP32 microcontroller, with physical reinforcement provided by a servo-motor barrier and driver-facing information delivered through an LCD display.

Evaluation through SUMO-based microscopic simulation demonstrated that this adaptive approach consistently and substantially outperforms fixed-time control. Reductions in average waiting time ranged from 34% under low traffic demand to 53% during peak congestion, and throughput gains reached 32% at the highest tested volume. These improvements are directly tied to the system's fundamental capability: dynamically reallocating signal time toward the lanes that need it most, rather than distributing it uniformly regardless of demand. The fact that these gains were achieved on hardware with a combined bill-of-materials cost well below that of conventional loop-detector installations underscores the practical viability of the approach for cost-sensitive deployment environments.

Two areas for near-term improvement emerged from prototype testing: the need for robust low-light detection capability and improved handling of heavily occluded queues. Beyond these refinements, the broader trajectory of the work points toward integration with cloud-based traffic management platforms and coordination across networks of intersections - a natural evolution that the ESP32's built-in Wi-Fi connectivity already partially enables.

X. FUTURE SCOPE

Several concrete directions merit investigation in follow-on work:

- Multi-Intersection Coordination: Connecting multiple intersection nodes via MQTT or emerging V2X protocols would allow the system to implement coordinated green-wave progression along arterial corridors, delivering network-level benefits that single-intersection optimisation cannot achieve.
- Predictive Control via LSTM: Incorporating an LSTM-based traffic flow predictor - as explored by Patel and Joshi [11] - would allow the system to preemptively adjust signal timings ahead of anticipated congestion rather than reacting after queues have already formed, particularly during predictable peak periods.

- Edge AI Optimisation: Deploying quantised and pruned versions of the detection model using TensorFlow Lite or Intel OpenVINO on dedicated neural processing units would substantially increase inference frame rates and reduce power consumption, enabling higher detection fidelity within the same embedded hardware budget.
- Enhanced Emergency and Priority Vehicle Management: Pairing the existing audio-based siren detector with visual siren-light recognition and integration with GPS-based dispatch systems would yield a far more robust and reliable priority mechanism than audio alone.
- Pedestrian and Cyclist Detection: Extending the detection pipeline to cover non-motorised road users would enable the system to schedule dedicated pedestrian and cyclist crossing phases in proportion to observed demand, improving safety and reducing unnecessary waiting for vulnerable road users.
- Cloud Dashboard and Remote Management: A web-based operations dashboard fed by real-time telemetry from each intersection node would give city traffic management centres visibility into system performance, the ability to override phases manually, and access to historical trend data for long-term planning.

REFERENCES

[1] World Health Organization, "Global Status Report on Road Safety 2023," WHO Press, Geneva, Switzerland, 2023.

[2] A. H. R. Tavanaei, A. Masoumi, and S. Mohammadi, "AI-Based Adaptive Traffic Signal Control: A Review," Journal of Intelligent Transportation Systems, vol. 25, no. 4, pp. 345-362, 2021.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 779-788, 2016.

[4] R. P. Roess, E. S. Prassas, and W. R. McShane, Traffic Engineering, 4th ed. Pearson Education, Upper Saddle River, NJ, USA, 2011.

[5] M. A. Chowdhury and A. W. Sadek, Fundamentals of Intelligent Transportation Systems Planning. Artech House, Boston, MA, USA, 2003.

[6] S. Jain and R. Verma, "Image Processing-Based Adaptive Traffic Signal Controller," in Proc. IEEE Int. Conf. Smart Transportation and Communication Networks, Wuhan, China, pp. 112-117, 2019.

[7] S. A. Patil, S. S. Shinde, and R. P. Patil, "Intelligent Traffic Signal Control System Using Image Processing and Vehicle Detection," in Proc. Int. Conf. Communication and Signal Processing (ICCSP), pp. 1120-1125, 2020.

[8] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018.

[9] Q. Li and Y. Wang, "Deep-Learning-Driven Dynamic Signal Timing for Urban Intersections," Transportation Research Part C: Emerging Technologies, Elsevier, vol. 115, pp. 1-18, 2020.

[10] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint arXiv:2004.10934, 2020.

[11] H. Patel and M. Joshi, "Traffic Flow Prediction Using LSTM Networks," in Proc. Springer Int. Conf. Smart Cities and Infrastructure, Singapore, pp. 301-312, 2021.

[12] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Real-time Tracking with a Deep Association Metric," in Proc. IEEE Int. Conf. Image Processing (ICIP), Beijing, China, pp. 3645-3649, 2017.

[13] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility," International Journal on Advances in Systems and Measurements, vol. 5, no. 3&4, pp. 128-138, 2012.

[14] S. Yin, Y. Yin, and W. Guo, "Real-Time Traffic Signal Control Based on Reinforcement Learning with Convolutional Neural Networks," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 6, pp. 3540-3550, 2021.

[15] F. Rasheed, K.-L. A. Yau, R. M. Noor, C. Wu, and Y.-C. Kwong, "Deep Reinforcement Learning for Traffic Signal Control Under Disturbances: A Case Study on Sunway City, Malaysia," Future Generation Computer Systems, Elsevier, vol. 109, pp. 604-619, 2020.