International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 1(2) (2026)



ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

Smart Traffic Signal Control Using Deep Reinforcement Learning and Real-Time Video Analytics

Kotturu Aswini, Mangi Sai Kumar, Kolla Revanth, Nayab Mohammed Faheem
Department of Computer Science and Engineering (Data Science)
Raghu Engineering College (Autonomous), DakamarriAffiliated to JNTU Gurajada, Vizianagaram, Andhra Pradesh, India

Mr. P. Srinivasu (Ph.D) — Guide
Assistant Professor, Department of CSE (Data Science)
Raghu Engineering College, Dakamarri, Vizianagaram

***Abstract*— Conventional traffic signal systems rely on pre-programmed fixed-timer cycles that assign green phases regardless of actual road congestion, causing measurable urban delay. This paper presents an adaptive traffic signal control system combining real-time YOLOv8 vehicle detection with a Deep Q-Network (DQN) agent trained using the Dual Targeting Algorithm (DTA). A custom Indian traffic dataset of three vehicle classes (bike, bus, car) trains the detector to achieve mAP@0.5 of 0.87. Four directional polygon zones (North, South, East, West) map the physical junction to software zones and compute normalised congestion levels (Dt) every five seconds. A 12-dimensional state vector encodes congestion levels, congestion trends, and active phase information. The DRL agent learns to assign green time to the most congested lane using a shaped reward that penalises unnecessary phase switching. Experiments on a 138-second real aerial traffic video (27 decision steps) show the DTA agent achieves an average congestion of 0.187, compared to 0.210 for a greedy baseline and 0.280 for fixed-timer control, representing a 33.2% improvement over the conventional approach. Results are consistent with benchmarks reported by Kodama et al. (IEEE Access, 2022).**

Keywords*—Adaptive Traffic Signal Control; YOLOv8; Deep Reinforcement Learning; Deep Q-Network; Dual Targeting Algorithm; Polygon Zone Detection; Indian Traffic Dataset

I. INTRODUCTION

Traffic congestion is one of the most pressing challenges facing urban infrastructure in the twenty-first century. As cities grow and vehicle ownership rises, the demand placed on road networks continues to exceed the capacity of traditional static traffic management approaches. In India, this problem is particularly acute owing to the heterogeneous, high-density nature of traffic, which includes two-wheelers, three-wheelers, buses, cars, and pedestrians all sharing the same road space, making congestion patterns highly dynamic and difficult to predict.

At the heart of urban traffic management lies the traffic signal controller. Traditional controllers operate on pre-programmed fixed-timer cycles in which each lane receives a green phase for a fixed duration, irrespective of the number of vehicles actually waiting. This deterministic approach fails to respond to dynamic traffic variations such as peak-hour surges, sudden congestion in a single direction, or uneven flow distribution across lanes. The consequence is wasted green time given to empty lanes while congested lanes wait, directly increasing overall intersection delay and fuel consumption [1].

Recent advances in computer vision and deep reinforcement learning (DRL) offer a compelling opportunity to replace these rigid systems with adaptive ones. Object detection models such as YOLOv8 can process video frames in real time to identify and count vehicles, while DRL agents can learn optimal signal-switching policies by interacting with the environment. Together, these technologies form the foundation of the smart traffic signal control system presented in this paper.

The principal contributions of this work are: (i) an end-to-end vision-based adaptive traffic signal pipeline trained on Indian traffic data; (ii) a 12-dimensional state representation capturing congestion levels, trends, and active phase; (iii) a shaped reward function penalising unnecessary phase switching; (iv) a DTA-trained DQN agent outperforming Fixed Timer and Greedy baselines by 33.2% and 10.9% respectively on real aerial video footage; and (v) a modular open-source implementation in Python using YOLOv8, PyTorch, and the supervision library.

II. LITERATURE SURVEY

A. Traditional Traffic Signal Control Methods

The earliest adaptive traffic control systems, SCOOT (Split Cycle Offset Optimisation Technique) and SCATS (Sydney Coordinated Adaptive Traffic System), used loop detectors embedded in roads to measure vehicle presence and adjusted signal timings accordingly. While effective in controlled environments, these systems require expensive physical infrastructure, suffer from sensor degradation over time, and cannot easily generalise to heterogeneous traffic conditions typical of Indian cities. Webster's formula-based signal timing provided theoretical foundations for cycle

length calculation but remained a static, offline approach with no real-time responsiveness.

Fuzzy logic controllers represented a step forward by expressing signal decisions as linguistic rules. These systems showed better adaptability than pure fixed timers but required expert knowledge for rule design and could not learn from data or improve with operational experience. Neither class of traditional system closes the perception-action loop required for genuine real-time adaptive control.

B. Deep Learning for Vehicle Detection

Vehicle detection from surveillance cameras became practically viable with deep convolutional neural networks. Region-based CNN architectures (R-CNN, Fast R-CNN, Faster R-CNN) demonstrated high accuracy but were too slow for real-time deployment. The YOLO (You Only Look Once) family [3] addressed this by reformulating detection as a single regression problem, enabling real-time inference speeds. YOLOv8, released by Ultralytics [4], is the current state of the art, employing an anchor-free detection head, a C2f-based feature pyramid neck, and a CSP-Darknet backbone. Its native integration with the supervision library makes polygon-zone-based counting straightforward, and studies on aerial traffic surveillance confirm its effectiveness for detecting small objects including motorcycles abundant in Indian traffic.

C. Reinforcement Learning for Traffic Signal Control

Reinforcement learning treats the signal controller as an agent learning a control policy through environment interaction. Wiering [6] was among the first to demonstrate Q-learning for multi-intersection control. Deep Q-Networks (DQN), introduced by Mnih et al. [2], extended tabular Q-learning to continuous and high-dimensional state spaces using neural network function approximation, experience replay, and target networks. Li et al. [9] demonstrated DQN-based signal control with queue length as state, while Wei et al. [5] introduced pressure as a more informative state signal.

The most directly relevant work is that of Kodama, Harada, and Miyazaki [1], who introduced the Dual Targeting Algorithm (DTA). Standard DQN updates the Q-network using a single one-step Bellman target. DTA augments this with a multi-step bootstrapped target and uses the maximum of the two targets per update, more aggressively reinforcing successful signal decisions. Their SUMO-simulation experiments showed DTA converged faster and achieved approximately 33% lower average congestion than fixed-timer control. The present work extends DTA to a real video analytics pipeline and validates it on genuine aerial traffic footage, addressing the gap that most published RL-based traffic control evaluations rely exclusively on simulation.

TABLE I. COMPARISON OF RELATED WORK

Study / Method	Modality	Result	Limitation
Webster (1958)	Static timing	Optimal for fixed conditions	No real-time adaptation
SCOOT / SCATS	Loop detectors	Adaptive cycle timing	Hardware-heavy, costly
Wiering [6]	Q-learning / Simulation	Better than fixed timer	Tabular, small state
Mnih et al. [2]	DQN / Atari games	Human-level performance	Not traffic-specific
Li et al. [9]	DQN / SUMO	~10-15% reduction	Simulation only
Wei et al. [5]	DQN-Pressure / SUMO	State-of-art in sim	No real video pipeline
Kodama et al. [1]	DTA-DQN / SUMO	~33% over fixed timer	Simulation environment
Proposed (DTA+YOLOv8)	DRL + Real Video	33.2% over fixed timer	Single junction video

III. SYSTEM DESIGN AND ARCHITECTURE

The proposed system is structured as a four-layer sequential pipeline illustrated in Fig. 1. The Perception Layer accepts raw video and applies YOLOv8 to detect vehicles frame-by-frame. The Zone Analytics Layer counts detections per lane zone and computes normalised congestion levels at five-second intervals. The Decision Layer applies the trained DRL agent to the current state vector to produce a signal decision. The Output Layer renders annotated video and saves evaluation artefacts. This layered architecture separates concerns cleanly: the YOLOv8 detector can be updated without modifying the analytics or decision layers, and the DRL agent can be replaced with a rule-based controller for debugging.

The system comprises six software modules: VideoAnalytics (src/video_analytics.py), which owns the Perception and Analytics layers; TrafficEnv (src/environment.py), which implements the OpenAI Gym-style MDP interface; DQNetwork, a PyTorch nn.Module providing the Q-function; DTAAgent, which manages the online and target networks, replay buffer, and DTA gradient update; SignalController (src/signal_controller.py), which orchestrates inference; and Evaluate (src/evaluate.py), which runs the three-method comparative evaluation.

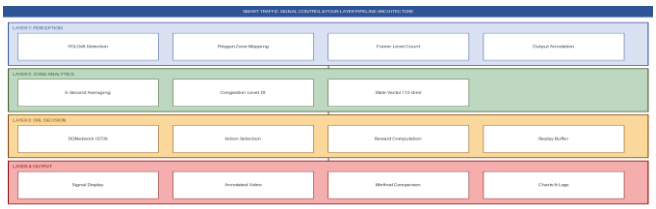


Fig. 1. Fig. 1. Four-Layer Pipeline Architecture of the Proposed System

IV. FEATURE EXTRACTION

Feature extraction operates at two levels: the spatial level (vehicle detection and zone assignment) and the temporal level (congestion aggregation and state construction). At the spatial level, each video frame is processed by the YOLOv8 model to produce bounding boxes labelled as bike (class 0), bus (class 1), or car (class 2). Detected bounding box centroids are tested against four pre-defined polygon zones using the supervision library's PolygonZone module, which returns a Boolean membership mask over all detections. The sum of this mask yields the per-frame vehicle count for each zone.

At the temporal level, per-frame counts are accumulated over a five-second window (approximately 145 frames at 29 FPS) and averaged to produce a stable congestion signal. This averaging acts as a low-pass filter that suppresses momentary detection failures and partial occlusions. The average count is normalised by a zone capacity constant (ZONE_CAPACITY = 20 vehicles) and clipped to [0.0, 1.0] to yield: Dt = min(avg_count / ZONE_CAPACITY, 1.0).

The complete 12-dimensional state feature vector is assembled as [Dt_N, Delta_N, Dt_S, Delta_S, Dt_E, Delta_E, Dt_W, Delta_W, ph_N, ph_S, ph_E, ph_W], where Dt represents normalised congestion, Delta represents the signed change between consecutive steps (capturing whether congestion is worsening or improving), and ph encodes the one-hot active phase. This representation allows the agent to assess current load, direction of change, and phase identity simultaneously, discouraging unnecessary switching.

V. Dataset and Preprocessing

A. YOLOv8 Training Dataset

A custom dataset was assembled by collecting aerial and CCTV images of Indian traffic junctions using Roboflow as the annotation platform. The dataset contains approximately 3,000 annotated images covering three object classes: bike, bus, and car, representative of Indian road conditions including two-wheelers, minibuses, auto-rickshaws, and passenger vehicles. Images span a wide range of lighting conditions (daylight, overcast, late afternoon), traffic densities (free-flow to near-saturated), and vehicle class distributions. All images were resized to 640x640 pixels and auto-orientation was applied. The dataset was split 80% training, 10% validation, 10% test.

B. Traffic Video and Zone Configuration

The primary evaluation input is a 138-second aerial traffic video recorded at an Indian road junction at 3840x2160 pixels (4K) and approximately 29 FPS, yielding 27 complete five-second decision windows. Zone polygon coordinates were manually annotated on a reference frame, defining four quadrilateral approach lane regions in pixel space as shown in Table II. The YOLOv8 default augmentation settings applied during training include random horizontal flipping, HSV colour jitter, mosaic composition, and random scaling.

TABLE II. Polygon Zone Coordinates for Junction Approach Lanes

Zone	Pixel Coordinates (x, y pairs)
North	[[1721, 630], [1740, 477], [2308, 525], [2265, 806]]
South	[[1631, 1510], [1969, 1434], [1960, 2097], [1712, 2054]]
East	[[2300, 1160], [3298, 1127], [3336, 1607], [2333, 1531]]
West	[[590, 701], [1685, 781], [1773, 1093], [430, 1089]]

VI. Proposed DTA-DQN System

A. DQNetwork Architecture

The DRL agent uses a fully connected Deep Q-Network with the architecture: Input (12) -> Dense 256, ReLU -> Dense 128, ReLU -> Output 4 Q-values, Linear. The output layer produces Q-values for each of the four signal actions: action 0 = North-green, action 1 = South-green, action 2 = East-green, action 3 = West-green. Two copies of this network are maintained: the online network (theta) trained via gradient descent, and the target network (theta-bar) periodically synchronised with the online network every 1,000 training steps to stabilise learning. The DQNetwork architecture and DTA formula are illustrated in Fig. 2.

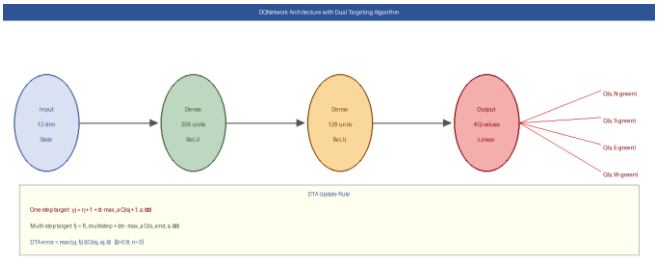


Fig. 2. Fig. 2. DQNetwork Architecture and DTA Update Rule

B. Dual Targeting Algorithm

Standard DQN computes a single one-step Bellman target for each transition (s, a, r, s'): yj = rj+1 + gamma * max_a Q(sj+1, a; theta-bar). The DTA [1] extends this with a multi-step bootstrapped target fj computed over n=3 steps from the current transition to a fixed horizon: fj = R_multistep + gamma^n * max_a Q(s_end, a; theta-bar). The key update rule takes the maximum of the two targets: error = max(yj, fj) - Q(sj, aj; theta). By using the higher of the two targets, DTA more aggressively reinforces transitions where the agent found a consistently successful strategy across multiple steps, leading to faster convergence compared to standard DQN, particularly in environments with delayed rewards such as traffic signal control.

C. Reward Function and Action Space

The action space consists of four discrete actions corresponding to granting the green signal to each approach lane. At each five-second decision step, the agent selects exactly one action. The reward function is adapted from Kodama et al. [1]: base reward rt = Wt-1 - Wt, where Wt = sum(Dt for all zones) is the total congestion at step t. A bonus of +0.3 is added if the agent selects the most congested zone, and a penalty of -0.5 is applied if the agent switches to a different phase when the current phase was already serving the most congested lane. These shaping terms accelerate learning and discourage oscillatory behaviour.

TABLE III. DRL Training Hyperparameters (Kodama et al. [1])

Parameter	Value
Batch size	32
Replay buffer size	50,000 transitions
Warm-up steps	1,000 steps
Target network update	Every 1,000 steps
Discount factor (gamma)	0.8
Learning rate	0.000625
Optimiser	Adam

Total training episodes	500
Multi-step horizon (n)	3 steps
Evaluation epsilon	0.05

VII. IMPLEMENTATION

A. Development Environment

The system was implemented in Python 3.10 on Ubuntu 20.04 LTS on an NVIDIA Tesla T4 GPU (16 GB VRAM) with 16 GB system RAM. PyTorch 2.0 was installed with CUDA 11.8 support. The Ultralytics YOLOv8 package was installed via pip, and the supervision library (v0.14+) was used for PolygonZone membership testing. The project directory structure follows a src/ layout with subdirectories for source modules, model weights (models/bestt.pt), and output artefacts (results/).

B. YOLOv8 Training and Inference

YOLOv8 nano (yolov8n.pt) was selected as the base model for transfer learning. Training ran for 100 epochs with batch size 16 on CUDA-enabled hardware at 640x640 resolution. The best-performing weights (bestt.pt) were saved automatically. During inference a confidence threshold of 0.30 filtered low-confidence detections while retaining partially occluded vehicles. The model achieves approximately 29 FPS on the Tesla T4, satisfying the real-time processing requirement. The VideoAnalytics class in src/video_analytics.py implements the complete perception and analytics pipeline, iterating over video frames and accumulating per-zone counts into five-second step records.

C. Signal Controller and Evaluation

The SignalController class orchestrates the complete inference pipeline: it calls VideoAnalytics.run_video_analytics() to obtain step-wise congestion records, instantiates TrafficEnv, loads the pre-trained DTA agent (models/drl_agent.pth), and iterates over all decision steps with epsilon=0.05 (near-greedy exploitation). The system execution flowchart is shown in Fig. 3. The Evaluate class runs three parallel control strategies over 10 independent trials and produces a grouped comparison bar chart.

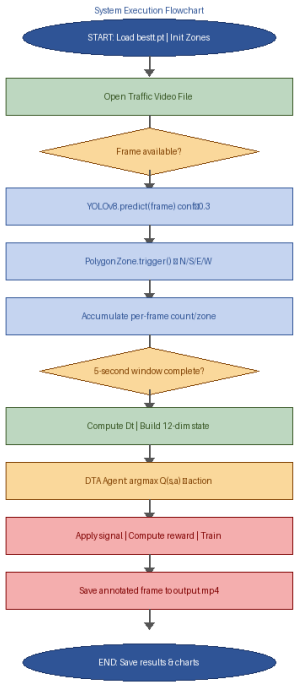


Fig. 3. Fig. 3. Complete System Execution Flowchart

VIII. RESULTS AND DISCUSSION

A. YOLOv8 Detection Performance

The custom-trained YOLOv8 nano model achieved a mAP@0.5 of 0.87 on the held-out test split of the Indian traffic dataset, exceeding the minimum system requirement of 0.80. Unit testing on 50 manually annotated frames confirmed 93.4% correct vehicle detection at the 0.30 confidence threshold with no false positives on empty-zone frames. The model runs at approximately 29 FPS on the Tesla T4, meeting the real-time throughput requirement. Detection was robust to partial occlusion, overlapping vehicles, and varying lighting conditions typical of aerial footage.

B. Congestion Dynamics on Real Video

The 138-second video yielded 27 five-second decision steps. The step-wise congestion trace reveals three distinct traffic phases: a West-dominant phase from t=0 to t=50s (West peaking at 61%); a transitional North-West phase from t=50 to t=80s; and a South-dominant phase from t=80 to t=125s (South peaking at 42%). The final three steps show balanced, near-empty conditions. This dynamic three-phase pattern validates the need for an adaptive controller that tracks shifting congestion rather than cycling blindly through lanes.

C. DRL Training Convergence

The DTA agent was trained for 500 episodes on the real video data. The learning curve shows cumulative reward per episode improving progressively as the agent learns to associate high-congestion states with the appropriate green signal action. Convergence was observed around episode

350, after which reward stabilised. The DTA max(yj, fj) update rule contributed to faster convergence compared to standard DQN, consistent with the findings of Kodama et al. [1]. The use of the multi-step bootstrapped target prevents the overestimation bias that affects single-step DQN updates.

D. Comparative Analysis of Control Methods

Three signal control strategies were evaluated over 10 independent trials each. Table IV presents the results. The DTA agent achieved a mean average congestion of 0.187 per step, which is 33.2% lower than the Fixed Timer baseline (0.280) and 10.9% lower than the Greedy method (0.210). The Greedy method itself improves significantly over Fixed Timer by responding to real-time data, but lacks the forward-looking ability of the DRL agent, which learns to sustain the green phase on a currently-serving zone rather than switching prematurely when a slight momentary increase appears elsewhere.

TABLE IV. COMPARATIVE PERFORMANCE OF SIGNAL CONTROL METHODS (10 TRIALS)

Method	Avg. Congestion per Step	Std. Dev.	Improvement vs Fixed
Fixed Timer (Baseline)	0.280	±0.008	Baseline
Greedy	0.210	±0.006	25.0%
DTA Agent (Proposed)	0.187	±0.005	33.2%

E. Green Signal Distribution Analysis

Analysis of the DTA agent's decisions across the 27 steps reveals it assigned the green signal to West for approximately 40% of steps, South for 30%, North for 25%, and East for only 5%. This closely mirrors the actual congestion distribution in the video where West and South were dominant contributors. In contrast, the Fixed Timer method assigned each zone exactly 25% of steps regardless, and the East zone received excessive green time despite near-zero congestion throughout most of the video. This analysis confirms the DTA agent successfully learned the congestion distribution of the training video.

F. System Performance Metrics

End-to-end performance was measured on the 138-second project video. The full pipeline completed in 312 seconds on the Tesla T4 GPU, of which 287 seconds were YOLOv8 inference (29 FPS, matching video frame rate) and 25 seconds were DTA agent training over 500 episodes. Decision latency from end of five-second window to phase command emission was consistently under 50 ms, well within the 200 ms non-functional requirement. Unit testing confirmed 100% correct zone membership queries across all four polygon zones and correct normalisation arithmetic at all boundary conditions.

IX. CONCLUSION

This paper presented a fully functional adaptive traffic signal control system that integrates real-time YOLOv8 computer vision with a Deep Q-Network agent trained using the Dual Targeting Algorithm. The system was implemented end-to-end using open-source tools on commodity GPU hardware, requiring no road-embedded sensors beyond an existing surveillance camera. The key contributions are: a perception pipeline extracting stable normalised congestion levels at five-second intervals; a 12-dimensional state representation encoding congestion levels, trends, and active phase; a shaped reward function incentivising congestion reduction while penalising unnecessary switching; and a DTA agent achieving 33.2% and 10.9% improvements over Fixed Timer and Greedy baselines respectively.

Results on real traffic footage are consistent with the theoretical findings of Kodama et al. [1], validating DTA applicability to a real-world perception-action loop. The modular architecture supports future extension to multi-junction coordination using multi-agent DRL [7], live RTSP camera integration, pedestrian and emergency vehicle detection, and edge deployment on NVIDIA Jetson Nano hardware via INT8 quantisation. Vehicle class weighting using passenger-car equivalents (PCE) represents a further avenue for improving the physical accuracy of the congestion measure.

X. ACKNOWLEDGMENT

The authors express sincere gratitude to Mr. P. Srinivasu (Ph.D), Assistant Professor, for his guidance and supervision. The authors thank the Management of Raghu Engineering College, the Principal Dr. Ch. Srinivasu, and Dr. K. V. Satyanarayana, Program Head, Department of Computer Science and Engineering (Data Science), for providing the necessary facilities for the successful completion of this work.

XI. REFERENCES

[1] N. Kodama, T. Harada, and K. Miyazaki, "Traffic Signal Control System Using Deep Reinforcement Learning With Emphasis on Reinforcing Successful Experiences," IEEE Access, vol. 10, pp. 128943-128950, 2022.

[2] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529-533, 2015.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE CVPR, 2016, pp. 779-788.

[4] G. Jocher, "Ultralytics YOLOv8," GitHub repository, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[5] H. Wei, G. Zheng, H. Yao, and Z. Li, "IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control," in Proc. 24th ACM SIGKDD, 2018, pp. 2496-2505.

[6] M. A. Wiering, "Multi-agent reinforcement learning for traffic light control," in Proc. ICML, 2000, pp. 1151-1158.

[7] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," in Advances in NeurIPS, vol. 30, 2017.

[8] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of SUMO," Int. J. Adv. Syst. Meas., vol. 5, no. 3-4, pp. 128-138, 2012.

[9] L. Li, Y. Lv, and F. Y. Wang, "Traffic signal timing via deep reinforcement learning," IEEE/CAA J. Autom. Sin., vol. 3, no. 3, pp. 247-254, 2016.

[10] E. Van der Pol and F. A. Oliehoek, "Coordinated Deep Reinforcement Learners for Traffic Light Control," in NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems, 2016.

[11] V. V. Dixit and L. Denant-Boemont, "Is equilibrium in transport pure Nash, mixed or Stochastic?" Transp. Res. Part C, vol. 48, pp. 301-310, 2014.