

International Journal Research Publication Analysis

Page: 01-11

TRINETRA: AI-DRIVEN INTELLIGENT ADAPTIVE TRAFFIC DIGITAL TWIN & OPTIMIZATION SYSTEM

***Namratha B. S., Chethan T. D., Harshith Kumar K., Gagan D., Vikas G.**

Dept. of Computer Science and Engineering Jyothy Institute of Technology Bengaluru, India.

Article Received: 18 September 2026

Article Revised: 08 October 2026

Published on: 28 October 2026

***Corresponding Author: Namratha B. S.**

Dept. of Computer Science and Engineering Jyothy Institute of Technology
Bengaluru, India.

DOI: <https://doi-doi.org/101555/ijrpa.9166>

ABSTRACT

Fixed-time traffic signals waste green time on empty approaches while vehicles queue on busy ones, and commercial adaptive systems are costly to instrument and hard to inspect. Existing learning-based controllers are rarely compared against a realistic actuated baseline, and often lack guaranteed safety rules. This paper reports the implementation of an adaptive signal controller for a four-way junction built on a SUMO digital twin. A Random Forest, trained on 38 simulation runs over 13 traffic scenarios, predicts per-lane vehicle count and waiting time 15 s ahead from 125 engineered features; it predicts the change over the horizon instead of the absolute level, with isotonic-calibrated confidence. A rule-based decision engine blends current and predicted demand into phase scores and enforces minimum and maximum green, hysteresis with a confirmation window, gap-out, starvation protection and emergency-vehicle priority. The controller is evaluated against vehicle-actuated control in two lockstep SUMO instances using seven metrics. The forecast reaches a test mean absolute error of 1.44 against 2.72 for a persistence baseline. The controller won or tied all seven metrics in all 13 scenarios, reducing average waiting time by 7.1% to 86.2%. A web console and an ESP32-driven four-head LED model show the live decisions. Results are from simulation only.

KEYWORDS: Adaptive traffic signal control, digital twin, random forest, SUMO, traffic prediction, vehicle-actuated control.

I. INTRODUCTION

Road junctions are shared resources: each second of green given to an empty approach is taken from one with waiting vehicles. The common way to share it is a fixed-time plan whose

cycle and splits are computed offline from average flows [1]. Such plans are simple and predictable but assume steady demand, so they cannot react when demand shifts, one approach overloads, a vehicle breaks down, or an ambulance arrives. Traffic in Indian cities adds a further difficulty, because cars, motorcycles, auto-rickshaws, buses and trucks share the same lanes with very different dynamics.

Actuated control lets detectors extend or end a green through minimum green, maximum green and gap-out rules [2], and adaptive systems such as SCOOT [3] and SCATS [4] adjust splits and offsets from detector data. These systems have a long field record but depend on dense detectors and calibrated models, and their decisions are hard to inspect. A digital twin, a virtual counterpart kept in step with the physical system [5], makes it possible to observe the junction, forecast its near future and test a controller safely before any field trial.

The unresolved problem is a controller that adapts to present and forecast demand, guarantees safety and fairness whatever the forecast says, explains every decision, and is evaluated honestly against a controller that gaps out like a real one. Learning-based controllers are usually compared with fixed-time or earlier learning baselines and rarely offer such guarantees; forecasting models are rarely tied to a control loop, and tree-based forecasters are rarely checked for their inability to extrapolate under overload.

The key contributions of this paper are as follows:

- We implement an end-to-end closed loop on a SUMO digital twin with strict module ownership and read-only side-channels, from sensing to a physical LED model.
- We train a Random Forest that predicts the 15 s change of per-lane demand with calibrated confidence, and show that residual targets cut the test error from 2.06 to 1.44.
- We design a transparent rule-based decision engine whose safety rules are independent of the forecast, and record a reason for every decision.
- We benchmark the controller against vehicle-actuated control in lockstep on 13 scenarios and seven metrics, and report the defects found and the cases that were lost.
- We drive an ESP32 model with four signal heads (16 LEDs) from the live decisions.

Section II reviews related work, Section III describes the methodology, Section IV the implementation, Section V the results, and Section VI concludes.

II. RELATED WORK

A. Fixed-time, Actuated and Adaptive Control

Webster's method [1] gives optimum cycle lengths and splits for steady flow, but the

resulting plan cannot respond to variation. Actuated control [2] adds demand responsiveness through gap-out, but sees only vehicles at the detector and has no view of arrivals. SCOOT [3] and SCATS [4] adapt plans in real time, and RHODES [8] adds prediction of arrivals and phase optimisation; all of them need detector coverage and calibrated network models and are difficult to inspect or modify.

B. Learning-based Control

Abdulhai et al. [9] applied Q-learning to a simulated junction, and IntelliLight [10] used a deep Q-network with real-world data from Jinan, China, outperforming fixed-time and earlier learning baselines. Their policies need long training, are hard to explain, and carry no built-in guarantee of minimum green, amber clearance or protection against starving an approach.

C. Traffic Prediction and Calibration

Lv et al. [11] used a stacked autoencoder on freeway data, and Li et al. [12] a diffusion-convolutional recurrent network on METR-LA and PEMS-BAY. They forecast flow and speed accurately but are not coupled to a controller and are heavy for a one-second loop. Random forests [13] are fast and accurate but cannot output values outside the range of their training targets. Niculescu-Mizil and Caruana [14] show how isotonic regression calibrates model outputs, which suggests a way of weighting a forecast by its reliability.

D. Research Gap

No surveyed work combines a learned 15 s forecast with a rule layer that guarantees safety and fairness, evaluates it against an actuated baseline on a wide scenario library, exposes an auditable reason for every decision, and closes the loop to physical lamps. The present study fills this gap.

III. PROPOSED METHODOLOGY AND SYSTEM ARCHITECTURE

Fig. 1 shows the end-to-end pipeline, which runs once per simulated second. SUMO produces raw state, which a traffic adapter converts into an immutable snapshot through TraCI [7]; a digital twin stores the state and a rolling history; a feature engineer produces 125 features; a Random Forest predicts 15 s ahead; a decision engine chooses the phase; and a signal controller applies it through a 3 s amber clearance. Database logging, a web console and the LED model observe the decision but cannot influence it.

Block Diagram of the Proposed System

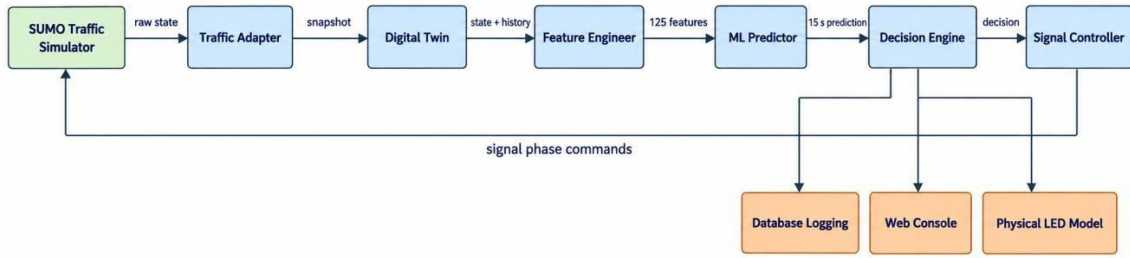


Fig. 1. Block diagram of the proposed system: every block has one job and one owner of the state it writes.

A. Scenarios and Dataset

The junction has four approaches of three lanes (left, straight, right), left-hand traffic, protected left turns and a vehicle mix of 55% cars, 30% motorcycles, 8% auto-rickshaws, 4% buses and 3% trucks. Thirteen scenario families (light, balanced, heavy, extreme, four directional-heavy, normal day, rush hour, rain, accident and emergency response) give 38 seeded runs. Each row pairs 125 features at time t with 24 targets (vehicle count and mean waiting time of 12 lanes) at $t + 15$ s. The data are split into 28,023 training rows, 6,514 chronological test rows and 4,802 held-out rows, the latter containing the out-of-distribution extreme scenario and held-out seeds of the accident and emergency scenarios.

B. Prediction Model

A multi-output Random Forest with 300 trees is used. A tree outputs the mean of the training rows in a leaf, so an absolute-level forest pulls long queues back into the training range. The model therefore predicts the change over the horizon $H = 15$ s and the current value is added back:

$$\hat{y}(t+H) = \max(0, y(t) + f(x(t))) \quad (1)$$

Where $x(t)$ is the feature vector and f the forest. The spread σ of the trees' outputs gives a raw confidence that is calibrated by an isotonic function g fitted on observed errors:

$$c = g(100 / (1 + \sigma)) \quad (2)$$

C. Decision Engine

The urgency of a lane l combines vehicle count n , mean waiting time $\bar{w}$ and maximum

waiting time $w_{\max}$, each normalised by a ceiling derived from recorded data:

$$S_{\text{cur}} = 0.60 \cdot n_N + 0.25 \cdot w_N + 0.15 \cdot w_{\max N} \quad (3)$$

Where the subscript N denotes a value normalised by its ceiling and capped at one. A predicted urgency S_{pred} is computed in the same way from the forecast, and the two are blended with a weight α that never exceeds a configured maximum and falls when confidence is low:

$$S = (1 - \alpha) \cdot S_{\text{cur}} + \alpha \cdot S_{\text{pred}} \quad (4)$$

The score of a phase aggregates the scores of its lanes and adds a soft starvation pressure that is capped below the switching margin. An ordinary scored switch from the current phase c to a rival j needs the rival to lead by a margin $m = 0.35$ for $\tau = 3$ s of consecutive time:

$$P_j - P_c > m \text{ for } t \geq \tau \quad (5)$$

Rules apply in a fixed order: emergency override, hard starvation (150 s), minimum green, gap-out (to the phase with most vehicles present, never by forecast), maximum green, a realistic-green floor of 20 s (12 s for right-turn phases) while the phase's own score is at least 0.12, and finally the scored switch. Every decision carries a mode and a reason.

D. Evaluation Design

Two SUMO instances of the same configuration file, one controlled by the AI and one by a baseline (fixed timer or vehicle-actuated control, VAC), are stepped in lockstep and measured by the same code. Each metric is a time-weighted

Algorithm

```

Algorithm 1: One decision tick
S(1) <- blend(Scur, Spred, alpha)
P(k) <- aggregate(S) + capped starvation
if emergency on another phase: switch
if phase unserved >= 150 s: switch
if green < mingreen: hold
if current empty and junction not:
    gap-out to phase with most vehicles
if green >= maxgreen: switch to best
if P(c) >= 0.12 and green < floor: hold
j <- argmax P over other phases
if P(j) - P(c) > m for tau seconds:
    switch to j
else: hold

```

V. RESULTS AND DISCUSSION

average and the signed improvement is for metrics where lower is better (and the reverse for speed and throughput). Seven metrics are used: average waiting time, average travel time, worst travel time, average and maximum queue length, average speed and throughput.

$$I = (M_B - M_A) / M_B \times 100\% \quad (6)$$

IV. IMPLEMENTATION

A. Hardware and Software Environment

Development and experiments ran on a Windows laptop without a GPU, with SUMO 1.27, Python 3.11, scikit-learn [15], FastAPI and uvicorn for the services, SQLite for logging, and a React and TypeScript console with three.js for the 3D view. The physical model uses an ESP32-WROOM-32 board with 16 addressable LEDs on GPIO 13, 4, 16 and 17 through 470 Ω resistors, powered from an external 5 V supply, with firmware written in the Arduino framework using the Adafruit NeoPixel library.

B. *Training Configuration*

Training and run time share one feature schema and the same adapter, twin and feature code, so training rows are built by the code that will serve the model. The model file records its target mode, and calibrators are fitted afterwards. Training used a chronological split with a held-out set that contains an unseen scenario.

C. *Engineering for Speed*

Reading five values of every vehicle at every 0.05 s step cost about 445,000 TraCI round trips per 60 simulated seconds. The adapter now reads state at the 1 Hz decision cadence from a one-shot fleet snapshot, which raised an extreme-traffic run from about 4.4 to about 11 times real time with byte-identical published outputs. The trained model is cached per process, so a console Start reaches its first tick in about 1.4 s.

D. *Deployment Pipeline*

A console server hosts a read-only WebSocket stream and a separate control layer; one live run executes on a worker thread. A hardware module maps the live signal state to the 16 lamps and sends one line per frame over USB serial at 115200 baud ('L' followed by 16 characters R, A, G or 0), repeated every second as a heartbeat. The firmware blinks the amber circles if no line arrives for 3 s so that a dead link never shows a stale green. A 0.5 s delay matches the lamps to the screen.

Prediction Accuracy

Table I and Fig. 2 compare the forecast with the persistence baseline (“nothing changes”) and a Random Forest with absolute targets. The residual model reduces the test error by 47% against persistence and by 30% against the absolute model, and its error on the out-of-distribution extreme scenario is 1.73. For the individual targets the test and held-out errors are 0.43 and 0.76 for vehicle count, and 2.45 and 2.21 s for waiting time.

TABLE I: PREDICTION ERROR (MAE) OF THE COMPARED MODELS.

Model	Test	Held-out
Persistence baseline	2.72	4.31
Random Forest, absolute target	2.06	2.35
Random Forest, residual target (proposed)	1.44	1.49

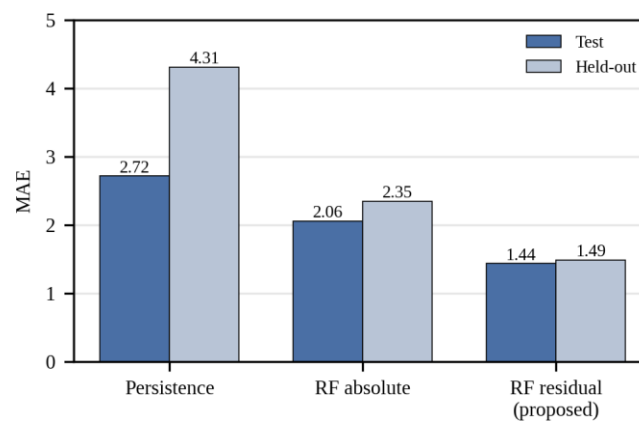


Fig. 2. Test and held-out MAE of the three predictors.

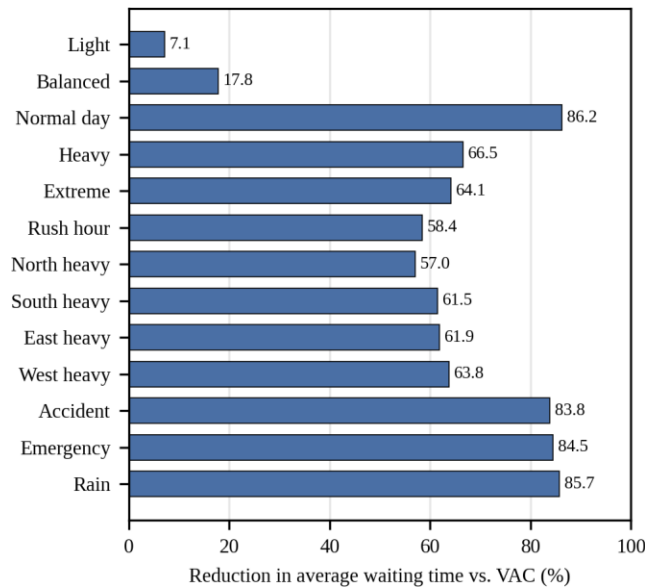
The gain comes from the target definition. A tree cannot emit a value above the training range, and the absolute model was worse than persistence on vehicle count on every lane. A second cause was a train/serve skew: the most important feature was a true countdown in training data but a re-armed 60 s ceiling at run time. Replacing it by an elapsed-time phase clock removed the skew without losing offline accuracy. The skew also showed that a forecast must be judged in the control loop: in light traffic the AI won 6 of 7 metrics with the forecast switched off but only 3 of 7 with the skewed model.

A. Control Performance Against VAC

Table II gives the improvement over VAC for five metrics in each scenario (seed 1). The controller wins or ties all seven metrics in all 13 scenarios; throughput is tied by construction because both controllers serve the same demand, and maximum queue is an exact tie in the light and balanced scenarios. Fig. 3 shows the waiting-time reduction.

TABLE II: IMPROVEMENT (%) OF THE AI OVER VAC, SEED 1.

Scenario	Wait	Travel	Worst	Queue	Speed
Light	7.1	0.3	7.8	0.7	1.6
Balanced	17.8	1.1	2.1	3.8	1.6
Normal day	86.2	34.7	70.3	65.2	46.9
Heavy	66.5	20.5	40.2	46.0	24.7
Extreme	64.1	37.3	57.4	51.5	45.5
Rush hour	58.4	10.8	43.8	25.9	10.9
North heavy	57.0	10.5	43.9	29.3	9.1
South heavy	61.5	8.8	42.4	26.4	9.5
East heavy	61.9	15.4	41.3	36.9	18.6
West heavy	63.8	14.8	43.1	37.6	16.1
Accident	83.8	34.0	70.2	62.5	41.3
Emergency	84.5	30.2	70.2	58.6	39.1
Rain	85.7	35.6	66.2	67.5	46.4

**Fig. 3. Reduction in average waiting time against VAC.**

The gain is smallest in light and balanced traffic, where an actuated controller that gaps out on empty approaches is already close to optimal, and largest where demand is heavy, uneven or disturbed: waiting time falls by 57% to 86% in the other 11 scenarios. Worst travel time improves by about 40% to 70% in those scenarios because the maximum-wait term, starvation protection and the green floor stop one approach being neglected. In the accident scenario the worst real journey was about 169 s under the AI against about 514 s under VAC, which has no starvation protection and kept feeding the blocked lane. In the three light seeds the AI switched less often than VAC (64 against 74, 58 against 67, 65 against 75), which supports the confirmation window as a flicker filter.

Not every design change helped. Lowering the starvation cap, or raising the minimum green for all phases, fixed the heavy east scenario but lost light traffic (3 of 7 to 4 of 7 on light seeds). The adopted floor applies only while the current phase still serves demand and leaves light traffic unchanged. The evaluation also found a defect: the evaluator passed no emergency lanes to the AI, so the emergency scenario had been measured with that logic off; after the fix it remained 7 of 7.

B. Runtime and Hardware

The two-sided extreme evaluation runs at 7.9 times real time unthrottled (0.9 before optimisation), and the full 13-scenario sweep takes about 6 minutes (about 75 before). The ESP32 identified itself at boot, completed a rig test of every phase and mirrored a live run, with East and West green while North and South were red at 35 s of simulated time.

C. Limitations of the Evidence

All results come from simulation on one synthetic junction. The decision parameters were tuned on the same scenario library, the sweep uses seed 1 of each scenario with a few extra seeds, and light seed 3 is only 5 of 7 (travel time -0.1% , speed -1.2%). The VAC baseline is our own implementation. The physical model was verified for function and timing, not benchmarked.

VI. CONCLUSION AND FUTURE WORK

This paper presented the implementation of an adaptive traffic signal controller that couples a Random Forest 15 s forecast with a rule-based decision engine on a SUMO digital twin. The forecast reaches a test MAE of 1.44 against 2.72 for persistence, and the controller wins or ties all seven metrics in all 13 scenarios against vehicle-actuated control, cutting average waiting time by 7.1% to 86.2%. A web console and a four-head ESP32 model make every decision visible.

The limitations are the simulation-only evidence, tuning on the evaluation library, a single junction and a self-implemented baseline. Future work will connect the adapter to real detector or camera data and calibrate against counts from a real junction; add a forecast of maximum waiting time; repeat the evaluation over many seeds with confidence intervals and against a learned controller; add pedestrian phases and a fail-safe fallback tested by fault injection; and measure end-to-end latency to the physical lamps.

ACKNOWLEDGMENT

The authors thank Mrs. Namratha B S, Department of Computer Science and Engineering, Jyothy Institute of Technology, for her guidance.

REFERENCES

1. F. V. Webster, "Traffic signal settings," Road Research Technical Paper No. 39, HMSO, London, U.K., 1958.
2. P. Koonce et al., "Traffic signal timing manual," Federal Highway Administration, Washington, DC, USA, Rep. FHWA-HOP-08-024, Jun. 2008.
3. P. B. Hunt, D. I. Robertson, R. D. Bretherton, and M. C. Royle, "The SCOOT on-line traffic signal optimization technique," *Traffic Eng. Control*, vol. 23, no. 4, pp. 190–192, Apr. 1982.
4. A. G. Sims and K. W. Dobinson, "The Sydney coordinated adaptive traffic (SCAT) system philosophy and benefits," *IEEE Trans. Veh. Technol.*, vol. 29, no. 2, pp. 130–137, May 1980.
5. M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems*. Cham, Switzerland: Springer, 2017, pp. 85–113.
6. P. A. Lopez et al., "Microscopic traffic simulation using SUMO," in *Proc. 21st IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, Maui, HI, USA, 2018, pp. 2575–2582.
7. A. Wegener et al., "TraCI: An interface for coupling road traffic and network simulators," in *Proc. 11th Commun. Netw. Simul. Symp. (CNS)*, Ottawa, ON, Canada, 2008, pp. 155–163.
8. P. B. Mirchandani and L. Head, "A real-time traffic signal control system: Architecture, algorithms, and analysis," *Transp. Res. C, Emerg. Technol.*, vol. 9, no. 6, pp. 415–432, Dec. 2001.
9. B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *J. Transp. Eng.*, vol. 129, no. 3, pp. 278–285, May 2003.
10. H. Wei, G. Zheng, H. Yao, and Z. Li, "IntelliLight: A reinforcement learning approach for intelligent traffic light control," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, London, U.K., 2018, pp. 2496–2505.
11. Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

12. Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, 2018.
13. L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
14. A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proc. 22nd Int. Conf. Mach. Learn. (ICML)*, Bonn, Germany, 2005, pp. 625–632.
15. F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.