International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Cognitive Traffic Flow AI: An Intelligent Traffic Monitoring, Congestion Analysis and Route Recommendation Framework

Authors: **Prashul Hegde, Pruthvi G, Thanvi R Shetty**
Students, AIML, VCET, Puttur, Karnataka, India
Corresponding author: Prof Venkatesh YC [yc.venkatesh@gmail.com]

Abstract

Vehicle counts alone describe road conditions poorly: a crowded carriageway may be flowing freely, while a handful of stationary vehicles can block it. This paper presents Cognitive Traffic Flow AI, a prototype decision-support framework that passes recorded traffic video through YOLOv8s detection and ByteTrack tracking, derives density, occupancy and image-space movement features, and labels each observation as Free Flow, Moderate or Congested with a multilayer perceptron whose output is then revised by movement rules. A Dynamic Congestion Index and a Traffic Risk Score, both scaled from 0 to 100 and defined within this project, add continuous severity information, and an LSTM forecasts seven traffic variables from 30-step windows. A Flask service connects these modules to a dashboard offering route planning (OpenRouteService, Nominatim, Leaflet and OpenStreetMap), locally generated explanations (Ollama) and e-mail alerts that fire only after sustained congestion. The classifier was assessed on 4,338 samples (3,138 video-derived from 22 videos and 1,200 synthetic) using five-fold grouped cross-validation. Mean accuracy was about 0.734 with a standard deviation of 0.167 across folds. Congested recall reached only 0.48, because 838 congested samples were labelled Moderate; we regard this as the principal weakness. The LSTM was trained on synthetic series, so its small errors show that the pipeline functions but do not demonstrate forecasting skill on real roads. Retraining with verified feature dimensions, speed calibration and validation on real data are identified as next steps.

Keywords: Traffic monitoring, YOLOv8, ByteTrack, Congestion detection, LSTM forecasting, Traffic risk assessment, Intelligent transportation systems, Decision support

1. INTRODUCTION

Congestion on urban and semi-urban roads is routinely observed through fixed cameras, and a considerable body of work has applied deep learning to turn that footage into congestion labels, short-term predictions or control decisions [1]. Single-stage detectors from the YOLO family [2], and in particular the current Ultralytics release [3], make per-frame vehicle detection practical on commodity hardware, while association-based trackers such as ByteTrack [4] keep vehicle identities stable across frames. Recurrent networks, notably LSTM [5], are a common choice when the quantity of interest is a time series.

Most deployed or prototyped pipelines nevertheless stop at one of these stages. A count of detected vehicles is easy to obtain but ambiguous: forty vehicles crossing a junction at speed and forty vehicles waiting behind a blockage give the same count. Distinguishing the two requires information about motion, about how much of the visible road is covered, and about how long an abnormal state has persisted. Operators additionally need the result in a usable form, namely a dashboard, a timely warning, a forecast and some guidance about travel, rather than a bare class label.

This paper describes Cognitive Traffic Flow AI, a prototype that couples the stages above in a single pipeline. We do not propose a new detector, tracker or forecasting network. The contribution is the integration and the engineering of the intermediate signals. Specifically:

- a movement-aware traffic-state pipeline in which tracked, image-space displacement is used to revise the output of a learned classifier;
- a continuous Dynamic Congestion Index and a Traffic Risk Score, both project-defined indicators on a 0–100 scale, together with a sustained-congestion rule that limits alert frequency;
- an LSTM forecasting module over seven traffic variables;
- a Flask-based dashboard that combines monitoring, forecasting, route display through open services, local language-model explanations and e-mail alerts; and
- a candid evaluation, including the weak recall for the Congested class and the synthetic nature of the forecasting data.

The remainder of the paper reviews related work, states the research gap, describes each module, reports the experiments and discusses limitations and future work.

2. PROBLEM STATEMENT AND OBJECTIVES

The problem addressed is the reliable estimation of road traffic state from video when vehicle count is an insufficient descriptor. A busy road with free-moving vehicles should not be flagged, whereas a modestly populated road with mostly stationary vehicles should. The system therefore has to reason jointly about count, density, occupancy, movement, persistence of the condition, expected evolution and the operational response.

The objectives are: (i) detect cars, motorcycles, buses and trucks with YOLOv8s; (ii) track them with ByteTrack and measure image-space movement; (iii) extract traffic features and classify the state into Free Flow, Moderate and Congested; (iv) compute a Dynamic Congestion Index and a Traffic Risk Score on 0–100 scales; (v) detect sustained congestion and send e-mail alerts; (vi) forecast near-future traffic variables with an LSTM; (vii) provide route display through OpenRouteService; and (viii) present all outputs,

together with a locally generated textual explanation, in a web dashboard.

3. RELATED WORK

3.1 Vision-based congestion detection

A survey of deep learning for congestion work [1] observes that traffic-camera images and video are the most frequently used sources for detection, and that convolutional models, YOLO among them, have been used to separate congested from non-congested scenes; in one camera-based study discussed there, binary classification accuracies above 90% are reported. More recent studies build directly on YOLOv8. Hari K.C. et al. [6] fine-tune the detector for video content analysis of congestion patterns, targeting categories such as heavy and normal congestion. A study published in JUTIF [7] combines YOLOv8 with ByteTrack to estimate flow and congestion in a monitored region and reports an accuracy of 80% for congestion detection. A further study [8] uses YOLOv8 to measure traffic density and flow on roads in Mauritius and reports a mean accuracy of 93.9% for traffic flow. These results come from different datasets, label definitions and metrics, so they cannot be compared numerically with each other or with the present work.

3.2 Multi-object tracking

Tracking-by-detection methods link per-frame detections into trajectories. SORT [9] relies on motion prediction and overlap; DeepSORT [10] adds appearance features; ByteTrack [4] retains low-confidence detections in a second association stage, which reduces identity loss in crowded scenes. Because movement in our system is computed from consecutive positions of the same identity, tracking stability directly affects the movement features.

3.3 Traffic forecasting

LSTM networks [5] have been applied to traffic speed and flow prediction, for example on microwave-sensor speed data [11] and in deep-learning approaches to flow prediction with large sensor datasets [12]. These studies rely on measured traffic series. Our forecasting module follows the same recurrent design but, in the experiments reported here, is exercised only on synthetic data.

3.4 Language models in transportation

Recent surveys describe how large language models are being examined across intelligent transportation systems, covering prediction, management and safety analytics [13], and note that natural language can act as an interface for interpretable human–machine interaction [14]. Both surveys also discuss reliability concerns such as unsupported or inconsistent generated statements. We use a locally hosted model only to verbalise values computed elsewhere.

TABLE 1. Representative studies and their emphasis

Ref.	Emphasis	Reported evidence (own metric)
[1]	Survey: deep learning for congestion detection, prediction, alleviation	Review; no single result
[6]	Fine-tuned YOLOv8 for congestion-pattern video analysis	Qualitative effectiveness stated
[7]	YOLOv8 + ByteTrack, flow and congestion in a region	80% congestion-detection accuracy
[8]	YOLOv8 density and flow on Mauritian roads	93.9% mean accuracy (traffic flow)

Metrics originate from different datasets and are not directly comparable.

4. RESEARCH GAP

Among the studies examined for this paper, the emphasis falls on individual components: detection and counting, congestion classification, or forecasting. Comparatively few of them describe an implementation that carries the result through to a continuous severity measure, a persistence-aware alert policy, a forecast, route display and a plain-language explanation in one operating dashboard. We make this statement only about the literature we reviewed; it is not a claim that no such system exists, and the review was necessarily selective. The present work addresses this integration gap with a prototype whose limitations are documented in Section 15.

5. PROPOSED FRAMEWORK AND SYSTEM ARCHITECTURE

Figure 1 shows the processing chain. Frames are read with OpenCV and resized to 1280×720 pixels. YOLOv8s detects vehicles and ByteTrack assigns identities. Tracked boxes yield counts, density, occupancy and movement. After standardisation, an MLP classifies the state and rule-based refinement can revise it. The Dynamic Congestion Index, Traffic Risk Score and congestion duration are then computed, an LSTM forecasts the next values, and a Flask backend exposes everything to the dashboard, the alert module, the route planner and the explanation service.

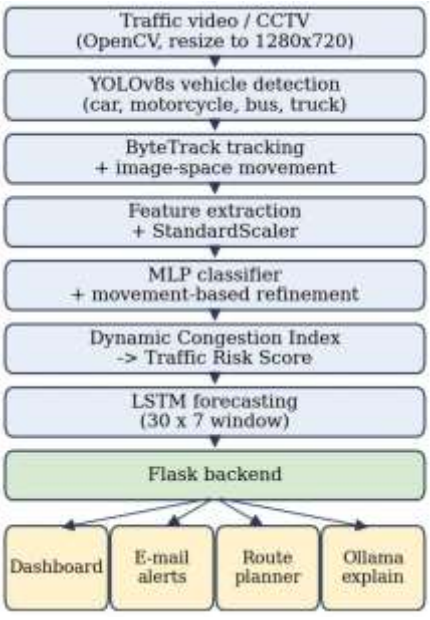


Fig. 1. Overall architecture of Cognitive Traffic Flow AI.

6. DATASET AND DATA PREPARATION

The most recent version of the classification dataset holds 4,338 samples. Of these, 3,138 are derived from real traffic video (22 distinct videos) and 1,200 are synthetic samples generated to cover controlled and edge conditions such as high density with little movement or high occupancy. Synthetic samples are not independent real-world observations and are reported separately throughout. Table 2 gives the composition and Table 3 the class distribution.

TABLE 2. Dataset composition

Source	Samples	Share
Real video-derived (22 videos)	3,138	72.3%
Synthetic	1,200	27.7%
Total	4,338	100%

TABLE 3. Class distribution (all samples)

Class	Samples	Share
Congested	1,947	44.9%
Moderate	1,203	27.7%
Free Flow	1,188	27.4%

Split of each class between real and synthetic samples.

Because several samples can originate from one video, random splitting would let near-identical scenes appear in both training and test partitions. Evaluation therefore uses StratifiedGroupKFold, which keeps all samples of a group in the same fold while preserving class proportions as far as possible. The grouping key used for synthetic samples .

7. VEHICLE DETECTION AND TRACKING

YOLOv8s (weights file yolov8s.pt) is applied to every processed frame, restricted to the COCO classes car (2), motorcycle (3), bus (5) and truck (7). Input frames are resized to 1280×720 pixels before detection and feature extraction; this is the processing resolution and not necessarily the native resolution of the source videos. The detector confidence threshold is 0.25. ByteTrack, used through the Ultralytics tracking interface, assigns persistent identities to detections; its configuration file and hyperparameters are the Ultralytics defaults.

For each tracked vehicle the centre of the bounding box is recorded, and the displacement between consecutive observations is taken as its movement (Eq. 5). The unit is pixels in the resized frame. **It is not a calibrated speed and must not be read as km/h;** it depends on camera placement, perspective and frame rate. Using these displacements, vehicles are grouped as moving, slow or stopped, the exact thresholds being .

The movement thresholds are defined in terms of pixel displacement between consecutive frames. A vehicle is classified as **moving** when its displacement is above the moving threshold, **slow** when it falls within the slow-motion range, and **stopped** when the displacement is below the stopping threshold. The exact threshold values used in the implementation are. These classifications are subsequently used as inputs to the traffic-state and congestion analysis.



Fig. 2. Sample output of detection and tracking.

8. TRAFFIC FEATURE EXTRACTION

Let V be the number of detected vehicles, W and H the frame width and height, and A = W×H the frame area in pixels. The density feature scales the count to one million pixels of frame area:

$$D = (V \times 1{,}000{,}000) / A \quad (1)$$

For vehicle i with box (x1, y1, x2, y2), the box area is $B_i = (x2 - x1)(y2 - y1)$. The average box-area ratio and the road occupancy ratio, the latter clipped to [0, 1] as in the implementation, are

$$R_{avg} = [(1/V) \sum_{i=1..V} B_i] / A \quad (2)$$

$$R_{occ} = min(max((\sum_{i=1..V} B_i) / A , 0), 1) \quad (3)$$

Short-term variability of the count is the standard deviation over a rolling history of up to n = 15 observations:

$$\sigma_V = \sqrt{[(1/n) \sum_{i=1..n} (V_i - V_{mean})^2]} \quad (4)$$

For a tracked vehicle with centre $C_t = (x_t, y_t)$, the image-space movement and the aggregate movement measures are

$$M_t = \sqrt{[(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2]} \quad (5)$$

$$R_{move} = N_{moving} / N_{total}, \quad M_{avg} = (1/V) \sum_{i=1..V} M_i \quad (6)$$

The ratio R_{move} is defined for $N_{total} > 0$; its value when no vehicle is tracked follows the implementation . The feature extractor currently emits five features: vehicle count, density D, average box-area ratio R_{avg}, count standard deviation σ_V and occupancy R_{occ}. Moving ratio and average movement are used by the refinement rules, the index tracker and the forecaster rather than as inputs of the classifier.

9. MLP-BASED CLASSIFICATION AND MOVEMENT REFINEMENT

Features are standardised as $z = (x - \mu)/\sigma$, with μ and σ estimated on training data only. A scikit-learn [15] MLPClassifier with two hidden layers is then trained for three classes (Table 4).

TABLE 4. MLP configuration

Parameter	Value
Library	scikit-learn MLPClassifier
Hidden layers	(16, 8)
Activation	ReLU
L2 penalty (alpha)	0.01
Maximum iterations	3000
Random state	42
Feature scaling	StandardScaler
Input dimension (verified)	5

Verification status of the input dimension. During development the extractor was extended with the occupancy ratio. The live pipeline passes this feature onward, but an earlier saved model artefact was reported to expect four inputs. Whether the final model and its scaler were retrained on the five-feature representation has not been confirmed. The number of inputs of the saved network, the feature count of the saved scaler and the feature order used in training and in live prediction must be checked before the results below are attributed to a five-feature model.

After the MLP predicts a label L_{ML}, two rules use the moving ratio. With $V_{min} = 6$, $R_{low} = 0.20$ and $R_{high} = 0.40$: if L_{ML} is Congested, $V \geq V_{min}$ and $R_{move} \geq R_{high}$, the final label becomes Moderate; if $V \geq V_{min}$ and $R_{move} \leq R_{low}$, the final label becomes Congested. The first rule prevents a dense but flowing scene from being reported as congested, and the second promotes dense, nearly static scenes. The reported metrics in Section 14 are for the classifier stage; whether the rules were applied to those predictions .

10. DYNAMIC CONGESTION INDEX AND TRAFFIC RISK SCORE

Dynamic Congestion Index (DCI). The DCI is a continuous indicator from 0 to 100. The tracker receives the vehicle count, road occupancy ratio, moving vehicle ratio and average movement, combines them in a project-specific function F, and smooths the result exponentially with $\alpha = 0.30$:

$$DCI_t = F(V_t, R_{occ,t}, R_{move,t}, M_{avg,t}) \quad (7)$$

$$DCI_{smooth,t} = \alpha \cdot DCI_t + (1 - \alpha) \cdot DCI_{smooth,t-1} \quad (8)$$

The DCI is a decision-support indicator specific to this project and has not been validated against an established traffic-engineering index.

Traffic Risk Score. The score also ranges from 0 to 100 and is mapped to four levels: LOW, MEDIUM, HIGH and CRITICAL. The implementation function takes the congestion index, vehicle count, moving ratio, congestion duration and occupancy:

$$Risk_t = G(CI_t, V_t, R_{move,t}, T_{cong,t}, R_{occ,t}), \quad 0 \leq Risk_t \leq 100 \quad (9)$$

Two sample outputs recorded in the project documentation illustrate the behaviour (Table 5). They are demonstrations, not validation. **The score is a heuristic severity indicator and must not be interpreted as a crash or accident probability.**

TABLE 5. Sample Risk Score outputs

Congestion index	Duration (s)	Occupancy	Score	Level
82	120	0.35	94	CRITICAL
5	0	0.02	3	LOW

Sustained congestion and alert rule. Duration T_{cong} grows by the elapsed time while the final state remains Congested and is otherwise reset or updated as implemented. An alert is eligible when $T_{cong} \geq 5$ s, and after an alert a further one is suppressed until at least 60 s have elapsed (Section 12).

11. LSTM-BASED TRAFFIC FORECASTING

At time t the model observes the seven-variable vector

$$X_t = [\, V_t, D_t, R_{occ,t}, R_{move,t}, M_{avg,t}, CI_t, Risk_t \,] \quad (10)$$

and a window of 30 consecutive vectors forms one input of shape 30×7. The network (Fig. 3) has two stacked LSTM layers with dropout, a dense layer and a seven-unit output giving the predicted next values of the seven variables. Table 6 lists the configuration. The 4,800-row series yields 4,770 windows (4,800 − 30); the first 80% (3,816) are used for training and the last 20% (954) for testing, chronologically. Inputs are scaled to [0, 1] with MinMaxScaler. The total parameter count of 32,135 is consistent with the layer sizes.

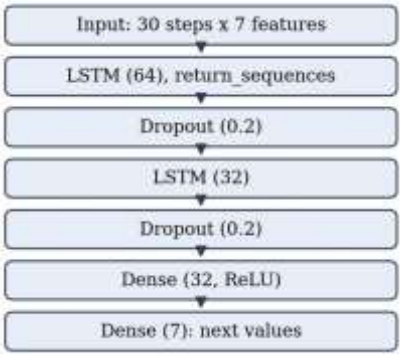


Fig. 3. LSTM forecasting network.

TABLE 6. LSTM configuration

Item	Setting
Framework	TensorFlow/Keras (2.21.0)
Input / target shape	(4770, 30, 7) / (4770, 7)
Train / test sequences	3,816 / 954 (chronological)
Layers	LSTM64, Drop 0.2, LSTM32, Drop 0.2, Dense32 (ReLU), Dense7
Parameters	32,135
Scaler	MinMaxScaler
Max epochs / batch	50 / 32
Early-stopping patience	10
Training data	Synthetic (4,800 rows)

12. ROUTE ASSISTANCE, EXPLANATION, BACKEND AND ALERTS

12.1 Route assistance

The user enters a start and a destination. Nominatim converts both to coordinates, OpenRouteService computes the route, and Leaflet draws it on OpenStreetMap tiles together with distance information. Route calculation is entirely the external engine's task; the classifier does not compute routes. No coupling between the monitored congestion state and route selection is claimed in this version, and Google Maps is not used.

12.2 Local explanation

Structured values (vehicle count, moving ratio, congestion index, risk score and traffic level) are passed to a locally

hosted language model through Ollama, which returns a short natural-language explanation. The model acts as an explanation layer and is not part of classification. The model used , and the correctness and usefulness of the generated text have not been evaluated.

12.3 Flask backend and dashboard

A Flask service exposes the endpoints /, /status, /video_feed, /alerts, /ai-explanation, /forecast, /route and /test-alert and links the vision pipeline, models, route services and front end. The dashboard has three pages: a route planner with map; a monitoring page showing per-class counts, total vehicles, moving ratio, average movement, congestion index, traffic level, confidence, risk score, congestion duration and forecast; and a page with the AI explanation, alert history and a test-mail button.

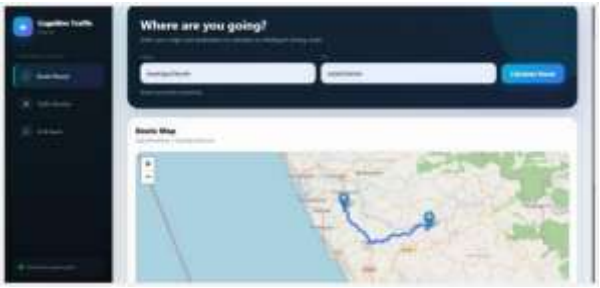


Fig. 4. Dashboard pages.

12.4 E-mail alerts

When congestion has persisted for the 5-second threshold, a message containing location, timestamp, traffic status and vehicle count can be sent through Gmail SMTP. A 60-second cooldown prevents repeated mails, and /test-alert verifies the mail path. Credentials are kept in an environment file rather than in source code.

13. EXPERIMENTAL SETUP

The classifier was evaluated with five-fold StratifiedGroupKFold. Precision, recall and F1 were computed per class as P = TP/(TP+FP), R = TP/(TP+FN) and F1 = 2PR/(P+R); accuracy is the proportion of correct predictions. For forecasting, mean absolute error and root mean squared error over n test points are

$$MAE = (1/n) \sum_{i=1..n} |y_i - \hat{y}_i| \quad (11)$$

$$RMSE = \sqrt{[\ (1/n) \sum_{i=1..n} (y_i - \hat{y}_i)^2\ ]} \quad (12)$$

Table 7 summarises the environment. Hardware and several software and tracker settings were not available when this manuscript was prepared and are marked for completion. Processing throughput was not formally benchmarked.

TABLE 7. Experimental environment

Item	Value
CPU / GPU / VRAM / RAM	Intel Core i5 / GPU optional / 16 GB RAM
Operating system / Python	Windows 11 / Python 3.11
TensorFlow	2.21.0
Ultralytics, OpenCV, scikit-learn versions	Ultralytics 8.3.0 / OpenCV 4.x / scikit-learn 1.x
Detector	YOLOv8s (yolov8s.pt)
Processing resolution	1280×720
Source resolution / FPS	Varies by source video
Detector confidence	0.25
ByteTrack settings	Default Ultralytics ByteTrack
Count-variability window	up to 15 observations
Sustained threshold / cooldown	5 s / 60 s

14. RESULTS AND EVALUATION

14.1 Cross-validation

TABLE 8. Five-fold grouped cross-validation accuracy

Fold	Accuracy
1	0.651
2	0.908
3	0.451
4	0.872
5	0.788
Mean	≈0.734
Std. dev.	≈0.167

Standard deviation is the population value over the five folds.

Fold accuracy ranges from 0.451 to 0.908, so performance depends strongly on which videos are held out (Fig. 5). A single mean would hide this spread. Per-fold class composition and the videos in each fold and would show whether the weak fold is dominated by a particular scene type.

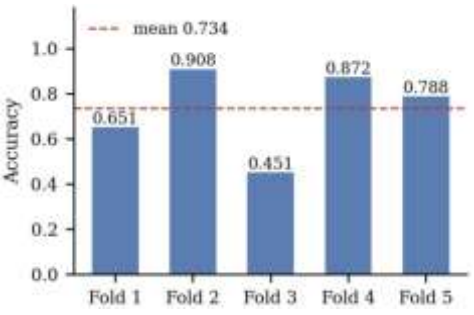


Fig. 5. Accuracy of each cross-validation fold.

14.2 Per-class performance and confusion matrix

TABLE 9. Per-class metrics

Class	Precision	Recall	F1
Free Flow	0.83	0.96	0.89
Moderate	0.56	0.92	0.70
Congested	0.95	0.48	0.64

TABLE 10. Confusion matrix (rows: true; columns: predicted)

True \ Pred.	Free Flow	Moderate	Congested
Free Flow	1143	20	25
Moderate	69	1109	25
Congested	171	838	938

Row sums equal the class totals in Table 3, so the matrix covers all 4,338 samples.

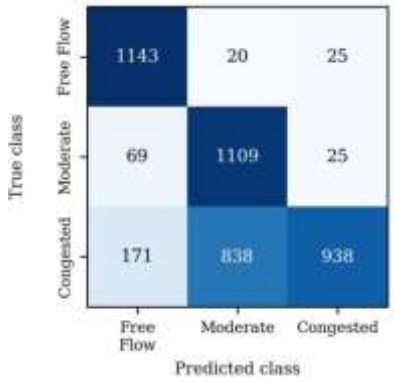


Fig. 6. Confusion matrix (Free Flow, Moderate, Congested).

The matrix reproduces the per-class figures: for example, Congested precision is 938/988 = 0.95 and recall 938/1947 = 0.48. Overall accuracy computed from the diagonal is (1143 + 1109 + 938)/4338 ≈ 0.735, in line with the fold mean. Whether the matrix pools out-of-fold predictions, and whether synthetic samples appear in test folds.

Congested recall is weak. Of 1,947 Congested samples, 938 (48%) were identified, 838 (43%) were labelled Moderate and 171 (9%) Free Flow. High Congested precision (0.95) together with high Moderate recall (0.92) and low Moderate precision (0.56) shows that the model is conservative about assigning the most severe label and that Moderate absorbs many severe cases. For a monitoring tool whose purpose is to flag severe congestion, this is the most important weakness of the present classifier.

14.3 Forecasting evaluation

TABLE 11. LSTM test errors (954 sequences)

Output	MAE	RMSE
Overall (7 variables)	0.8235	1.4616
Congestion index	0.0033	0.0040

These values come from a synthetic series and are not evidence of real-world forecasting accuracy. They show that the windowing, training and inference chain runs end to end. The very small congestion-index error, on an index whose nominal range is 0–100, suggests that the synthetic series is smooth and highly predictable or that the errors are expressed in scaled units; the unit of the reported errors . No naive baseline such as persistence was evaluated, so the improvement over trivial prediction is unknown.

15. DISCUSSION AND LIMITATIONS

The results support a limited conclusion: the integrated pipeline operates and produces interpretable outputs, but the learned congestion classifier is not yet reliable for the severe class. Several explanations for the Congested-to-Moderate confusion are plausible and should be tested rather than assumed. The five classifier inputs describe how many vehicles are present and how much area they cover but contain no movement information, so dense flowing traffic and dense stopped traffic may look similar to the network; movement is introduced only by the later rules. Label boundaries between Moderate and Congested may be partly subjective. Differences between videos, and the mixture of real and synthetic data, may also shift class boundaries between folds, as the spread from 0.451 to 0.908 suggests. An ablation comparing the MLP alone, the MLP with refinement rules and a classifier that includes movement features would show which factor matters; none of these was run.

Further limitations are as follows. (1) Evaluation uses recorded video, not a large live multi-camera deployment. (2) Movement is in pixels, not calibrated speed. (3) The saved MLP input dimensionality is unverified (Section 9). (4) The dataset includes 1,200 synthetic samples. (5) The LSTM was trained and tested on synthetic data and compared with no baseline. (6) The DCI and Risk Score are project-defined, their internal formulas are not reproduced here, and neither has been validated; the Risk Score is not a crash probability. (7) Only stopped or slow-vehicle and congestion reasoning is part of the system; pedestrian, animal, wrong-way and accident detection were not trained or evaluated. (8) Routing and geocoding depend on external services, and explanation quality depends on the chosen local model, which was not evaluated. (9) Adaptive frame sampling was considered conceptually but not benchmarked, so no speed-up is claimed. (10) No statistical significance or confidence analysis, and no comparison with baseline classifiers, was performed.

16. FUTURE WORK

Priority items are: confirm or redo MLP training with a verified five-feature (or extended) input and report its metadata; add movement features to the classifier and run the ablations above; collect larger real rural and semi-urban datasets with documented labelling; improve severe-congestion recall, for instance through class weighting or threshold tuning, and report per-fold results; calibrate cameras to obtain physical speed; validate the forecaster on measured series against persistence and other baselines; benchmark throughput and adaptive frame sampling; validate or recalibrate the Risk Score against recorded incidents; evaluate the generated explanations; and extend to RTSP/CCTV streams, multi-camera operation, edge deployment and, only after separate training and testing, additional incident classes.

17. CONCLUSION

Cognitive Traffic Flow AI combines YOLOv8s detection, ByteTrack tracking, movement-aware classification, two project-defined 0–100 indicators, LSTM forecasting, open-service route display, local explanations and sustained-congestion e-mail alerts within a Flask dashboard. On 4,338 samples the classifier reached a mean grouped cross-validation accuracy of about 0.734, but with wide variation between folds and a Congested recall of only 0.48. The forecasting results rest on synthetic data and demonstrate functionality only. The prototype therefore establishes feasibility of the integrated design, while real-world

validation, verified model dimensions and improved severe-congestion detection remain the main tasks ahead.

ACKNOWLEDGMENT

The authors thank their guide, Prof. Venkatesh YC, Assistant Professor, VCET, Puttur.

REFERENCES

[1] N. Kumar and M. Raubal, "Applications of deep learning in traffic congestion detection, prediction and alleviation: A survey," Transportation Research Part C, 2021; arXiv:2102.09759.

[2] 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.

[3] G. Jocher, A. Chaurasia and J. Qiu, "Ultralytics YOLOv8," software, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[4] Y. Zhang et al., "ByteTrack: Multi-object tracking by associating every detection box," in Proc. ECCV, 2022; arXiv:2110.06864.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] Hari K.C., M. Pokharel and S. Shrestha, "The prediction of road traffic congestion patterns: An optimized YOLOv8-based approach," in Proc. 3rd Int. Electronic Conf. on Processes, MDPI, 29–31 May 2024.

[7] M. M. Fahrezi and E. A. Laksana, "Traffic flow and congestion detection with YOLOv8 and ByteTrack," JUTIF, vol. 5, no. 4, Aug. 2024, doi: 10.52436/1.jutif.2024.5.4.2063.

[8] S. Pudaruth, I. M. Boodhun and C. W. Onn, "Reducing traffic congestion using real-time traffic monitoring with YOLOv8," Int. J. Advanced Computer Science and Applications (IJACSA), vol. 15, no. 10, 2024.

[9] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple online and realtime tracking," in Proc. IEEE ICIP, 2016, pp. 3464–3468.

[10] N. Wojke, A. Bewley and D. Paulus, "Simple online and realtime tracking with a deep association metric," in Proc. IEEE ICIP, 2017, pp. 3645–3649.

[11] X. Ma, Z. Tao, Y. Wang, H. Yu and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," Transportation Research Part C, vol. 54, pp. 187–197, 2015.

[12] Y. Lv, Y. Duan, W. Kang, Z. Li and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," IEEE Trans. Intelligent Transportation Systems, vol. 16, no. 2, pp. 865–873, 2015.

[13] D. Mahmud, H. Hajmohamed, S. Almentheri, S. Alqaydi, L. Aldhaheri, R. A. Khalil and N. Saeed, "Integrating LLMs with ITS: Recent advances, potentials, challenges, and future directions," IEEE Trans. Intelligent Transportation Systems, pp. 5674–5709, 2025.

[14] T. Nie, J. Sun and W. Ma, "Exploring the roles of large language models in reshaping transportation systems: A survey, framework, and roadmap," arXiv:2503.21411, 2025.

[15] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Machine Learning Research, vol. 12, pp. 2825–2830, 2011.