Indian Journal of Computer Science and Technology
Volume 5, Issue1 (January-April 2026), PP: 392-395.
www.indjcst.com **ISSN No: 2583-5300**

Implementation of a Smart City Traffic Flow and Signal Optimization

P. Gavaskar[*1], K. Govindharaj[*2], M. Ajay Kumar[*3]
[1]*Assistant Professor, Department of Information Technology, PSV College of Engineering and Technology, Krishnagiri, Tamil Nadu, India.*
[2,3]*UG Scholars, Department of Information Technology, PSV College of Engineering and Technology, Krishnagiri, Tamil Nadu, India.*

To Cite this Article: *P. Gavaskar[*1], K. Govindharaj[*2], M. Ajay Kumar[*3], "Implementation of a Smart City Traffic Flow and Signal Optimization", Indian Journal of Computer Science and Technology, Volume 05, Issue 01 (January-April 2026), PP: 392-395.*

Copyright: ©2026 This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution License; Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract: *The rapid expansion of urban mobility has intensified the need for intelligent and dynamic traffic management systems which are proficient to respond for real- time situations. This study presents Intelli Flow, a novel framework for predictive traffic pattern optimization in smart cities, driven by the proposed Deep Q- Network with Genetic Algorithm Flow Optimizer (DQN-GA Flow). The hybrid model integrates deep reinforcement learning and evolutionary computation to achieve optimized signal control and efficient traffic flow management. The DQN component enables dynamic state-action mapping for continuous learning of traffic conditions, while the genetic optimization mechanism refines policy parameters to enhance convergence stability and global search performance. Simulation experiments were conducted using a benchmark traffic dataset within a controlled smart city environment, and comparative analysis was performed against three existing algorithms: Fuzzy Logic-Based Traffic Controller (FLTC), Q- Learning-Based Traffic Signal Control (QL-TSC), and Deep Deterministic Policy Gradient (DDPG). The evaluation metrics included Average Vehicle Delay, Network Throughput, Convergence Rate, and Average Travel Time. The results demonstrated that DQN-GA Flow achieved significant improvement in performance, recording reductions of up to 24% in travel time and 22% in vehicle delay, along with notable enhancements in throughput and learning efficiency. The findings confirm that hybrid reinforcement learning combined with genetic optimization provides an effective approach for achieving adaptive, data-driven, and sustainable traffic management in smart city infrastructures.***

I. INTRODUCTION

Traffic management plays a crucial role in the development of smart cities, where increasing population and rapid urbanization have led to heavy vehicle density on roads. Traditional approaches such as manual monitoring and CCTV surveillance are not sufficient for handling modern traffic complexities. These methods require large manpower, result in delayed responses, and often fail to identify real-time violations like overspeeding. Therefore, cities require advanced automated systems capable of continuous monitoring and timely intervention to ensure smooth and safe transportation.

With the advancement of Artificial Intelligence and Computer Vision technologies, intelligent transportation systems have become a practical solution to improve traffic regulation. Automated detection and analysis of vehicles help authorities better understand road usage and act immediately against rule violations. The integration of AI-driven systems supports decision-making and traffic planning, reducing accidents and congestion. These systems are a major component of the smart city concept where technology and data improve overall urban living.

The proposed Smart City Traffic Management System utilizes Python, Flask web framework, and YOLOv8 (You Only Look Once, version 8) deep learning model to automate various traffic monitoring tasks. YOLOv8 enables accurate and fast object detection, allowing the system to identify different vehicle types in real time. Combined with frame-by-frame tracking, the movement of each vehicle can be precisely monitored to calculate speed and count traffic flow.

Literature Review

Advanced Learning for ITS

This survey examines the role of advanced machine learning technologies-such as deep learning, reinforcement learning, transfer learning, and federated learning-in intelligent transportation systems (ITS).[1] It reviews applications including traffic prediction, autonomous driving, incident detection, and smart signal control. The authors discuss challenges like data privacy, model robustness, scalability, and real-time processing constraints.[18] The paper also highlights emerging trends such as edge AI and cooperative learning among vehicles and infrastructure.

CNN for City Traffic Prediction

This work proposes a convolutional neural network (CNN)-based approach for short-term city traffic prediction. Traffic data is represented as spatial grids, allowing CNNs to extract spatial dependencies effectively.[2] The model achieves high accuracy compared to traditional and other deep learning methods.[17] The authors demonstrate that CNNs can handle complex, high-resolution traffic datasets and support real-time forecasting. This contributes to improving traffic management and reducing congestion in urban environments.

II.METHODOLOGY

The methodology of the Smart City Traffic Management System involves multiple stages that work together to provide accurate and real-time traffic surveillance. The process begins with video input acquisition from IP cameras, CCTV cameras, or pre-recorded traffic footage. These video streams are fed into the system where frame-wise analysis is performed continuously to detect and track passing vehicles.The core engine of the system is the YOLOv8 deep learning model, which is responsible for detecting vehicles in each frame. YOLOv8 is trained on diverse datasets, allowing it to recognize different vehicle types like cars, buses, motorcycles, and trucks with high precision. The model processes each frame quickly enough to meet real-time performance requirements, even in crowded traffic scenes.

After detection, the system performs vehicle tracking to ensure continuous monitoring of each detected vehicle across the video frames. A unique ID is assigned to every vehicle so that its movement can be recorded without confusion. Tracking algorithms such as SORT or DeepSORT are used to match objects between frames based on position and motion patterns, enabling accurate trajectory mapping.

Once tracking is established, the system proceeds with vehicle speed estimation. Speed is calculated based on pixel displacement and time duration between detections. To ensure accuracy, a real-world reference distance must be calibrated within the frame. This distance is measured as a known meter value between two visible points in the camera view. The camera remains fixed to maintain consistent calibration throughout the monitoring period.

Speed Calculation (Practical Steps)

Calibrate a real-world distance (D_real):Measure a fixed distance in meters between two visible reference points in the camera view.

Measure the pixel distance (D_pixels):Find the distance between the same two points in the video frame in terms of pixels.

Record vehicle positions at two different timestamps (t_1 and t_2): Extract pixel coordinates of the tracked vehicle (unique ID assigned) at both timestamps.Compute pixel displacement:$\Delta p = p_2 - p_1$\Delta p = p_2 - p_1Δp=p2−p1

Convert pixel displacement into real-world

distance:$distance_m = \Delta p \times \left(\frac{D_{real}}{D_{pixels}} \right)$distance_m=Δp×(DpixelsDreal)

Calculate speed in meters per

second:$speed_{m/s} = \frac{distance_m}{(t_2 - t_1)}$speedm/s=(t2−t1)distance_m

Convert to kilometer sperhour: $speed_{km/h} = speed_{m/s} \times 3.6$\text{speed}_{km/h} = \text{speed}_{m/s} \times 3.6speedkm/h=speedm/s×3.6

This mathematical approach eliminates the need for costly physical speed detectors such as radar or laser sensors. It enables highly scalable deployment using only security cameras and software processing.

The system also includes vehicle counting, which tallies the number of vehicles passing through the camera's region. This data provides valuable insights into traffic flow patterns. Both traffic volume and speed records are displayed on a web dashboard and stored in a database for future analysis, such as congestion prediction, infrastructure planning, and monitoring peak traffic times.The entire workflow is integrated through a Flask web application, where the backend performs detection, tracking, speed calculation, and alert generation. The output is provided in real time through a dashboard interface, displaying live video with bounding boxes, current speed readings, and alert notifications for speed violations. Database management ensures long-term storage and easy retrieval of traffic reports.

Finally, continuous performance testing is conducted to enhance system reliability under different lighting and weather conditions. Model thresholds, speed limits, and camera calibration values can be adjusted to improve adaptability across various monitoring locations. This methodology ensures that the system remains accurate, efficient, and scalable for smart city traffic automation.

4.1 System Configuration (Hardware & Software)

The Smart City Traffic Management System requires an efficient combination of hardware and software components to ensure accurate vehicle detection, speed calculation, and smooth real-time performance. The hardware configuration is selected to support continuous video processing, while the software stack provides AI-based functionalities, web deployment, and database handling. On the hardware side, a high-performance CPU such as an Intel i5/i7 or AMD Ryzen equivalent is essential for handling rapid video frame processing.

Although the system can run without a GPU, incorporating a dedicated graphics card like NVIDIA GTX/RTX greatly improves YOLOv8 model performance and enhances real-time processing capability, especially under high-traffic scenarios. At least 8GB RAM is recommended to maintain stable system operations, and storage of minimum 256GB SSD ensures fast data access and storage for recorded traffic logs.In addition to computing hardware, the system utilizes IP cameras or CCTV cameras installed at road intersections or highways. These cameras provide live video streams at 720p or 1080p resolution, ensuring clear visibility of vehicles for precise detection and tracking. A reliable network connection is crucial for transmitting video feeds to the server without frame drops, enabling accurate speed calculations and alert generation.

For the software component, the system is powered by Python as the primary programming language due to its strong support

for machine learning and computer vision libraries. The Flask web framework is used to build the interactive dashboard and manage backend communication, allowing browser- based access to real-time monitoring. Flask ensures simplicity, scalability, and faster deployment of application services.The vehicle detection module relies on YOLOv8, a state-of-the-art deep learning model. YOLOv8 is trained using advanced neural network architectures capable of detecting vehicles in varying lighting and weather conditions. The model makes use of libraries such as PyTorch**,** OpenCV**,** and NumPy for high-speed image processing and object recognition. These libraries allow accurate bounding box placement and movement tracking for speed estimation.

To store and manage system data efficiently, databases like MySQL or SQLite are integrated. The database maintains essential information such as timestamp, vehicle type, speed reading, and violation logs. This enables long-term analysis of traffic trends and allows authorities to review records at any time. The storage structure is optimized for fast retrieval and report generation.

The system also utilizes web technologies like HTML, CSS, and JavaScript for designing a user-friendly interface. The dashboard displays live video feed overlays, vehicle count statistics, speed indicators, and alert messages when a violation occurs. The smooth UI design ensures that traffic monitoring personnel can quickly interpret data and take immediate actions.For deployment, the system can run on local servers or cloud platforms such as AWS, Google Cloud, or Microsoft Azure, making it suitable for centralized monitoring across multiple city locations. Cloud integration also provides secure backup, remote accessibility, and better infrastructure scaling as traffic volume increases.

In summary, the combination of robust hardware, cutting-edge AI models, and a scalable web architecture ensures that the Smart City Traffic Management System performs efficiently under real-time operational conditions. The system is future-ready and supports easy upgrades as traffic monitoring needs continue to grow in smart cities.

III. MODELING AND ANALYSIS

- The existing traffic monitoring systems implemented in many urban environments still rely heavily on traditional manual observation.
- Although CCTV cameras are widely deployed at intersections, highways, and important city routes, the analysis of this video feed is usually done by human operators.
- This manual monitoring process is slow and lacks accuracy due to human fatigue, limited attention span, and delayed responses.
- Another major limitation of the existing systems is the dependency on physical hardware sensors such as radar guns and speed detectors.
- These devices need to be installed at multiple points across the city, which demands high cost, frequent maintenance, calibration, and skilled technicians.
- The existing systems also lack real-time processing capabilities. In many cases, video footage is recorded and reviewed later only when an accident occurs or when authorities need evidence.
- As there is no automated mechanism to detect overspeeding or violation patterns instantly, authorities cannot intervene immediately to prevent further incidents.
- Installing more sensors increases cost, maintenance issues, and points of failure. Moreover, manually managing hundreds of cameras becomes nearly impossible for human operators. This makes the system unsustainable for large-scale deployment in rapidly expanding urban environments.

To overcome these limitations, the proposed Smart City Traffic Management System introduces an AI-based automated solution using YOLOv8 for real-time vehicle detection and Flask for web-based monitoring. The system detects vehicles in camera footage, calculates their speed, counts total traffic flow, and generates alerts if speed limits are exceeded. This automation reduces human involvement and greatly enhances accuracy and response speed.

The proposed system operates continuously, even in conditions where human operators may not be available. It can identify multiple vehicles simultaneously with high precision, ensuring efficient monitoring in high-traffic locations. Its ability to track each vehicle individually enables accurate speed calculation and violation tracking without requiring any physical sensors embedded on the roads.

A key improvement in this system is its centralized dashboard, allowing traffic officials to monitor multiple locations remotely via a single web application. All collected data such as speed details, number of vehicles, and violation logs are stored in a database, providing long-term insights that can be used to plan infrastructure improvements and optimize traffic flow.

The proposed system supports scalability and can be integrated with additional smart modules, such as automated number plate recognition (ANPR), digital fine payment systems, and cloud-based analytics. This flexibility makes it suitable for deployment in different road environments, from small intersections to major highways.

Another major advantage of the proposed system is cost-effectiveness. By using open- source tools like Python, OpenCV, and YOLOv8, the system avoids expensive specialized hardware.

IV. RESULTS AND DISCUSSION

The system also performs automated vehicle counting, allowing traffic flow analysis and congestion detection. All extracted information—speed, counts, violation history—is stored in a structured database for long-term analytics and policy-making.A user-friendly Flask web dashboard displays live video streams, vehicle detection boxes, vehicle counts, overspeed alerts, and statistical charts. Traffic officers can monitor multiple camera feeds from a single centralized platform, eliminating the need for physical presence at traffic junctions.The complete system is cost-effective, uses open-source tools, and is designed for easy scaling across multiple intersections, highways, and public places. Optional modules such as ANPR (Number Plate Recognition)**,** automated fine payment, or cloud-based analytics can be integrated without changing the core architecture.

V. CONCLUSION

The Smart City Traffic Management System using Python and YOLOv8 represents a significant advancement in intelligent transportation solutions. The system successfully demonstrates how artificial intelligence and computer vision can automate vehicle monitoring while improving road safety and efficiency. By providing real-time detection, speed calculation, and violation alerts, the application reduces reliance on human supervision and enhances enforcement accuracy.The integration of advanced deep learning models like YOLOv8 highlights the importance of AI in current and future traffic management systems. Another major achievement of the system is its ability to function with ordinary CCTV/IP cameras, eliminating the need for expensive hardware. This makes the solution not only technologically superior but also economically feasible for widespread implementation across smart cities, especially in developing countries.

The centralized dashboard powered by Flask enhances operational efficiency by presenting all traffic monitoring results in a clear and user-friendly format. Real- time alert notifications ensure that traffic enforcement teams can take immediate actions to prevent dangerous driving conditions and ensure safer roads for the public. In conclusion, the Smart City Traffic Management System is a robust, automated, and intelligent solution that enhances the safety and efficiency of urban transportation. With future enhancements such as ANPR, smart signal integration, and violation-based automation, this system has strong potential to evolve into a fully-featured smart traffic ecosystem, contributing greatly to modern smart city development.

REFERENCES

1. Chen, Q. et al. (2019) ‘A Survey on an Emerging Area: Deep Learning for Smart City Data’, IEEE Transactions on Emerging Topics in Computational Intelligence, 3(5), pp. 392–410
2. Ashokkumar, C. et al. (2024) ‘Urban Traffic Management for Reduced Emissions: AI-based Adaptive Traffic Signal Control’, Proceedings of the 2nd International Conference on Sustainable Computing and Smart Systems (ICSCSS), Coimbatore, India, pp. 1609–1615.
3. Alruban, A. et al. (2024) ‘Artificial Hummingbird Optimization Algorithm With Hierarchical Deep Learning for Traffic Management in Intelligent Transportation Systems’, IEEE Access, 12, pp. 17596– 17603.
4. Chen, Q. et al. (2019) ‘A Survey on an Emerging Area: Deep Learning for Smart City Data’, IEEE Transactions on Emerging Topics in Computational Intelligence, 3(5), pp. 392–410.
5. Chen, L. et al. (2021) ‘Data-Driven C-RAN Optimization Exploiting Traffic and Mobility Dynamics of Mobile Users’, IEEE Transactions on Mobile Computing, 20(5), pp. 1773–1788