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

Smart Traffic Management System Using Machine Learning

KHUSHI MISHRA
M.Sc. Information Technology (Part II)
D. G. Ruparel College of Arts, Science and Commerce, Mumbai, India

Abstract: Urban traffic congestion is worsened by fixed-timer signals that give every lane the same green time regardless of demand, and the resulting delay is most dangerous for emergency vehicles such as ambulances and fire trucks. Existing priority solutions often depend on extra hardware such as RFID tags, GPS transmitters or in-road sensors, while many camera-based systems only count vehicles and do not recognize emergency vehicles. This paper presents the design of a low-cost, vision-based Smart Traffic Management System that uses camera video of a four-way intersection to detect and count vehicles with a YOLO object detector, identify emergency vehicles with an AlexNet-based classifier trained on custom images, estimate per-lane congestion, and adjust green and red signal timings automatically. In normal operation, green durations follow real-time congestion within minimum and maximum limits so that no lane is starved. When an emergency vehicle is confirmed, its lane is turned green after a short safety interval while the other lanes are held at red, and the normal cycle is restored once the vehicle has passed. The system is built with open-source tools (Python, OpenCV, Ultralytics YOLO and Tkinter) and is designed to run on an ordinary computer. The paper also sets out an evaluation protocol that compares the adaptive system with a fixed-timer baseline on the same video, and justifies each design choice using published benchmarks of the underlying models.

Keywords: Smart Traffic Management, Machine Learning, YOLO, AlexNet, Emergency Vehicle Detection, Adaptive Signal Control, Computer Vision.

I. Introduction

Traffic congestion is a growing problem in cities, and most traffic signals still operate on fixed timers that give every lane the same green time whatever the number of vehicles waiting. Such signals cannot react to real-time demand, which leads to long waiting times, wasted fuel and unnecessary queues at intersections.

The problem is more serious for emergency vehicles such as ambulances and fire trucks. When they are stuck in a dense queue, they depend on sirens and on other drivers making way, and a delay of even a few minutes can cost lives and property. Manual control by traffic police is not possible at every junction.

Advances in computer vision and deep learning make it possible to use ordinary traffic cameras as sensors. Real-time object detectors such as YOLO can detect and count vehicles from video, and convolutional neural networks such as AlexNet can classify cropped vehicle images. This paper proposes a Smart Traffic Management System that combines these techniques to optimize green light durations according to lane congestion and automatically give priority to emergency vehicles, using only camera input and no additional road or vehicle hardware.

The main contributions are: a camera-only pipeline that integrates vehicle detection, emergency vehicle identification, congestion estimation and signal control in one low-cost software system; an emergency priority module that clears the path for an emergency vehicle and then restores the normal signal cycle; and an evaluation protocol that compares the adaptive system with a fixed-timer signal on the same test videos.

The rest of the paper is organized as follows. Section II states the objectives, Section III reviews related work, Section IV describes the methodology, Section V discusses the design in light of published benchmarks, Section VI lists limitations, Section VII outlines future scope, and Section VIII concludes the paper.

II. Objectives

The general objective is to design, implement and evaluate a low-cost, camera-based traffic signal control system that reduces vehicle waiting time at a four-way intersection and automatically provides a clear path for emergency vehicles. The specific objectives are:

1. To capture traffic video of each lane using OpenCV, with frame resizing and frame skipping so that processing runs smoothly on an ordinary computer.
2. To detect and count vehicles (car, bike, bus, truck) in real time using a YOLO-based detector, counting each vehicle once as it moves through the lane.
3. To identify emergency vehicles (ambulance, fire truck) using YOLO-based tracking together with an AlexNet-based classifier trained on custom emergency vehicle images.
4. To estimate the congestion level of each lane (low, medium or high) from the vehicle count.
5. To develop an ML-based decision algorithm that optimizes green light durations according to congestion, within minimum and maximum limits so that no lane is starved.
6. To implement an emergency priority module that, after a short safety interval, turns the emergency vehicle's lane green and the others red, and restores the normal cycle after the vehicle has passed.
7. To build a Tkinter interface that shows live video, vehicle counts, congestion levels, signal status and emergency alerts.
8. To evaluate the system using accuracy, precision and recall, and to compare average waiting time and emergency vehicle clearance time with a fixed-timer signal.

III. Literature Review

A. Fixed-Time and Sensor-Based Adaptive Signal Control

Webster proposed the classic method of calculating fixed-time signal settings, which remains widely used. The Sydney Coordinated Adaptive Traffic System (SCATS) introduced adaptive control driven by road sensors. Adaptive systems of this kind respond to demand but depend on sensors installed in the road, which are costly to install and maintain.

B. Object Detection

YOLO frames detection as a single regression problem solved in one network pass. On PASCAL VOC 2007, the base model reached 63.4% mAP at 45 frames per second, and the smaller Fast YOLO reached 52.7% mAP at 155 frames per second on a Titan X GPU, making real-time video processing possible. YOLOv4 later reported 43.5% AP (65.7% AP50) on MS COCO at about 65 frames per second on a Tesla V100 GPU.

Detectors of this family are commonly trained on MS COCO, which contains 91 common object categories in about 328,000 images with 2.5 million labeled instances. Its categories include vehicles such as car, bus, truck and motorcycle, but not ambulances or fire trucks, so custom images are needed for emergency vehicle recognition.

C. Image Classification

AlexNet has five convolutional and three fully connected layers with about 60 million parameters. It won the ILSVRC-2012 competition with a top-5 error of 15.3% against 26.2% for the runner-up, which established deep convolutional networks as the leading approach to image classification. Its compact design makes it suitable for classifying cropped vehicle images as normal or emergency vehicles.

D. Learning-Based Signal Control

IntelliLight applied deep reinforcement learning to traffic light control and tested it on a large-scale real traffic dataset obtained from surveillance cameras, whereas earlier studies had not used real traffic data. Reinforcement learning, however, requires extensive training from traffic data, which motivates a lighter approach based on congestion estimation from camera video.

E. Tools

OpenCV provides free tools for video capture and image processing, and the Ultralytics YOLO framework provides pre-trained detection models, which together make a low-cost implementation practical.

F. Research Gap

Many existing systems either optimize normal traffic flow or give emergency priority using extra hardware, and camera-based systems often count vehicles without recognizing emergency vehicles. Priority systems usually do not consider the congestion of the other lanes. This work addresses the gap by combining camera-based congestion handling and emergency vehicle priority in one low-cost software system.

IV. Methodology

The system processes the video of each lane through a sequential pipeline: video input, vehicle detection and counting, emergency vehicle identification, congestion estimation, signal timing decision, emergency priority control, and display on a user interface.

A. Requirement and Data Definition

The vehicle classes, intersection layout and lane views are defined first. Traffic videos of a four-way intersection with one camera view per lane are collected, together with emergency vehicle images from public sources. Only vehicle types are detected; no individuals are identified and number plates are not read, so privacy risk is low.

B. Video Input

Video of each lane is captured with OpenCV, from recorded or live sources. Frames are resized and selectively skipped to reduce computation so that the system runs on an ordinary computer.

C. Vehicle Detection and Counting

A lightweight pre-trained YOLO model from the Ultralytics framework detects cars, bikes, buses and trucks in every lane. Detected vehicles are tracked across frames so that each vehicle is counted only once as it moves through the lane.

D. Emergency Vehicle Identification

Tracked vehicles are cropped and passed to an AlexNet-based CNN that classifies each as a normal vehicle or an emergency vehicle (ambulance or fire truck). Running the classifier only on cropped vehicle images, rather than on the whole frame, reduces computation. Because MS COCO does not contain emergency vehicles, the classifier is trained on a custom image set collected from public sources, with data augmentation and transfer learning used to compensate for its limited size. To avoid false alarms, a confidence threshold and confirmation over several consecutive frames are applied before a vehicle is declared an emergency vehicle.

E. Congestion Estimation

The congestion level of each lane is calculated from its vehicle count and categorized as low, medium or high.

F. Signal Timing Decision

An ML-based decision algorithm sets the green light duration of each lane according to its congestion level. The duration is bounded by a minimum and a maximum green time so that no lane is starved and no lane monopolizes the intersection.

G. Emergency Priority Control

When an emergency vehicle is confirmed in a lane, the system waits for a short safety interval, then turns that lane green and holds all other lanes at red, creating a clear passage. Once the vehicle has crossed the stop line, the normal adaptive cycle is resumed.

H. User Interface

A Tkinter interface lets the operator select the video source and shows the live video, per-lane vehicle counts, congestion levels, signal states and emergency alerts. Signals are shown in a software simulation; the system is not connected to physical traffic signal hardware.

I. Development Environment

The system is developed in Python on Windows 10/11 using OpenCV, Ultralytics YOLO, TensorFlow/Keras or PyTorch, NumPy and scikit-learn, with VS Code or PyCharm and Git/GitHub as development tools. All are free and open source. The target machine is an ordinary computer (Pentium i3 processor, 4 GB RAM).

J. Evaluation Protocol

The system is evaluated on the same test videos against a fixed-timer baseline using four measures: detection and classification accuracy, precision and recall for normal and emergency vehicle classes; average waiting time per vehicle; emergency vehicle clearance time, defined as the time from first detection to the vehicle crossing the stop line; and processing speed in frames per second on the target hardware.

V. Discussion

This section explains why the chosen design is expected to meet its goals, using figures published by the original authors of the underlying models. These figures were measured on the authors' own datasets and GPUs and are not measurements of the proposed system.

Real-time detection. YOLO was chosen because it processes video in a single network pass. The original YOLO reported 45 frames per second and Fast YOLO 155 frames per second on a Titan X GPU, and YOLOv4 reported about 65 frames per second on a Tesla V100. These GPUs are far more powerful than the target machine, which is why the proposed system uses a lightweight model, reduced resolution and frame skipping, and why its own frame rate must be measured on the target hardware.

Emergency vehicle classification. MS COCO covers 91 categories with 2.5 million labeled instances but has no emergency vehicle class, so a separate classifier is needed. AlexNet reached a 15.3% top-5 error on 1.2 million training images across 1000 classes. The emergency task is a much smaller binary decision on cropped vehicle images, so a compact network is appropriate, although the small custom dataset makes augmentation and transfer learning essential.

Camera-only control. Sensor-based adaptive control such as SCATS requires road installations, and reinforcement learning approaches such as IntelliLight require heavy training. A camera-only approach avoids installation and maintenance costs and uses free software, which makes later real-world adoption more affordable. Bounding green times by minimum and maximum limits keeps the intersection fair to all lanes, and confirmation over several frames reduces the chance that a false emergency detection causes unnecessary priority and congestion in other lanes.

The system is designed as a decision-support simulation and does not disturb any existing traffic arrangement. Its measured accuracy, waiting-time change and clearance-time gain will be obtained by running the evaluation protocol in Section IV-J.

VI. Limitations

- Emergency vehicles are identified visually only; siren sound is not used.
- Accuracy may drop in poor lighting, at night, in heavy rain or fog, or when vehicles block each other, and depends on camera angle and position.
- A lightweight YOLO model and reduced resolution are used for speed, which may slightly reduce accuracy compared with larger models.
- The emergency class relies on a small custom image set of limited size and variety. Ambulance and fire truck designs differ between regions, and unusual or unmarked vehicles may be misclassified.
- The system is a software simulation and is not connected to physical traffic signal hardware; results on test videos may not fully match real roads with unpredictable driver behaviour.
- Pedestrian crossings, non-motorized traffic and the coordination of multiple intersections are not handled.
- Performance depends on the computer used; a low-end machine may need a lower frame rate.

VII. Future Scope

- Connecting the system to real traffic signal controllers using IoT hardware.
- Adding siren sound detection to improve emergency vehicle identification.
- Coordinating multiple intersections along an emergency route.
- Improving performance at night and in rain or fog.
- Using reinforcement learning for signal timing and adding pedestrian crossing support.

VIII. Conclusion

This paper presented the design of a low-cost, camera-based Smart Traffic Management System that detects and counts vehicles using YOLO, identifies emergency vehicles using an AlexNet-based classifier, adjusts green light durations according to lane congestion, and gives an automatic clear path to emergency vehicles before restoring normal operation. By relying only on camera video and open-source software, it avoids the cost of in-road sensors and vehicle-mounted hardware.

Although the current implementation is a software simulation with the limitations noted above, it provides a foundation for deployment with real signal controllers and for extensions such as siren detection, multi-intersection coordination and reinforcement-learning-based timing.

References

[1] F. V. Webster, “Traffic signal settings,” Road Research Technical Paper No. 39, Her Majesty’s Stationery Office, London, U.K., 1958.

[2] P. R. Lowrie, “The Sydney Coordinated Adaptive Traffic System: Principles, methodology, algorithms,” in Proc. IEE Int. Conf. Road Traffic Signalling, 1982.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, real-time object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 779–788.

[4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal speed and accuracy of object detection,” arXiv:2004.10934, 2020.

[5] T.-Y. Lin et al., “Microsoft COCO: Common objects in context,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2014, pp. 740–755, doi: 10.1007/978-3-319-10602-1_48.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Adv. Neural Inf. Process. Syst. (NIPS), 2012, pp. 1097–1105.

[7] H. Wei, G. Zheng, H. Yao, and Z. Li, “IntelliLight: A reinforcement learning approach for intelligent traffic light control,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD), 2018, pp. 2496–2505.

[8] G. Bradski, “The OpenCV Library,” Dr. Dobb’s Journal of Software Tools, 2000.

[9] Ultralytics, “Ultralytics YOLO documentation.”

Available: https://docs.ultralytics.com

[10] OpenCV, “OpenCV documentation.”

Available: https://docs.opencv.org

[11] Python Software Foundation, “tkinter Python interface to Tcl/Tk.”

Available: https://docs.python.org/3/library/tkinter.html