# Dynamic Traffic Light Management System

**Anup Murumkar[1], Danish Shaikh[2], Sanskriti Sharma[3] , Dhiraj Khivasara[4]**

*[1,2,3,4] Student, Department of Electronics and Telecommunications Engineering, Vishwakarma Institute of Information Technology, Pune, Maharashtra, India*

---------------------------------------------------------------------***---------------------------------------------------------------------

**Abstract -** *The increase in urban population and vehicle numbers has resulted in a critical issue: traffic congestion. This issue not only frustrates and delays drivers, but also contributes to higher fuel use and air pollution. While pollution impacts many locations, megacities face the brunt of the consequences. To properly manage this ever-increasing difficulty, real-time road traffic density must be monitored to ensure efficient signal control and traffic management. The effectiveness of traffic control has a substantial impact on traffic flow, necessitating optimization to meet increasing demand. Our suggested approach uses live video feeds from traffic junctions to calculate traffic density using image processing and artificial intelligence (AI). Furthermore, our system includes an algorithm that adjusts traffic lights based on vehicle density, to reduce congestion and improve transit times, and reducing pollution.*

***Keywords:*** **Traffic control, Traffic light system, Traffic management, Intelligent transport systems, Smart surveillance, Computer Vision, Machine Learning, Object detection, YOLO.**

## 1. INTRODUCTION

Urban regions are dealing with an increase in vehicle traffic, resulting in capacity restrictions on road networks and a lower level of service. One important contributing issue is the use of fixed signal timers at crossings, which do not adjust to changing traffic circumstances. This rigidity causes several traffic-related concerns. As the demand for road capacity increases, there is an urgent need for innovative traffic control technologies, notably in the field of Intelligent Transport Systems (ITS).

Mumbai and Bangalore are prime examples of how severe traffic congestion can be. Bangalore has the terrible title of having the world's worst traffic flow, closely followed by Mumbai in fourth place, as per a comprehensive report on traffic conditions across numerous cities and countries. During rush hours, journeys in Bangalore take 71% longer, while in Mumbai, the increase is 65% [9].

Currently, three main methods are used for traffic control:

1) **Manual Control**: This system requires human intervention, usually from traffic police stationed in specified regions. They use signboards, signal lights, and whistles to control traffic flow.

2) **Traditional Traffic Lights with Fixed Timers**: These traffic lights use predetermined timers, with fixed numerical values defining the duration of red and green signals. These timed settings cause the lights to go on and off automatically.

3) **Electronic Sensors**: A more advanced solution involves installing loop detectors or proximity sensors on highways. These sensors collect real-time traffic data, which is utilized to change traffic signal timings accordingly.

These conventional approaches have several limitations. The manual controlling system demands a significant amount of labor. Due to limited traffic police resources, human traffic management is not feasible in all regions of a city or town. Thus, a more effective traffic control system is required. Static traffic control involves a fixed traffic signal with a timer for each phase, which does not respond to real-time traffic on the road. Using electronic sensors, such as proximity sensors or loop detectors, can lead to conflicting accuracy and coverage due to the expense of collecting high-quality information. Limited budgets may limit the number of facilities available. Furthermore, due to the limited effectiveness range of motion, the total coverage on a network of facilities requires a lot of sensors.

Dynamic traffic management control systems have been prompted by the advent of new technology, especially in the fields of computer vision and artificial intelligence (AI). These systems optimize traffic flow, reduce congestion, and improve overall road network efficiency by utilizing the capability of real-time data collecting and analysis. This study examines the theory and practice of a dynamic traffic management control system, which aims to upend established traffic control models. This system attempts to dynamically modify traffic signal timings based on real-time traffic density and vehicle categorization data by utilizing computer vision techniques and clever algorithms. By using cutting-edge technology like YOLO (You Only Look Once), the suggested system is expected to yield notable enhancements in traffic control effectiveness, resulting in decreased fuel consumption and environmental impact.

## 2. Literature Review

**Reference [1]** This paper presents a deep Q-learning-based reinforcement learning (RL) strategy to adaptive traffic signal regulation at junctions. To optimize traffic flow, the model uses real-time traffic data (queue lengths and waiting times) and adjusts traffic signal phases accordingly. Simulation findings illustrate the potential for decreased congestion and greater efficiency when compared to typical fixed-time controls.

**Reference [2]** This research presents an intelligent traffic signal management system based on real-time traffic flow and ultrasonic sensors. The technology monitors vehicle presence in each lane and modifies the green light length appropriately, favoring lanes with the highest traffic flow. The authors note possible benefits such as reduced congestion and increased traffic flow efficiency. However, drawbacks include a failure to account for elements such as pedestrian traffic and the possibility of failing sensors.

**Reference [3]** This paper presents a unique Multi-Agent Reinforcement Learning (MARL) technique for Adaptive Traffic Signal Control (ATSC) in an Internet of Things (IoT) context. The system relies on real-time traffic data acquired by IoT sensors (surveillance cameras) at crossings. The important features are: 1) Distributed MARL using an Advantage Actor-Critic (A2C) approach that employs Deep Neural Networks (DNNs) for policy learning at each intersection. 2) Information transmission between nearby junctions improves coordination and accounts for larger traffic flow impacts.
The technique is tested on a real-world traffic network in Shiraz, Iran, which simulates six linked crossings. The results indicate that traffic flow might be enhanced when compared to the current fixed-time control scheme

**Reference [4]** Traffic signal timing is critical for efficient traffic movement. Here are two major approaches: 1) Traffic Engineering: Develops timing plans for single or connected crossings utilizing techniques such as Webster (minimizes travel time) and GreenWave (creates progression bands). Sensors can also be employed to implement adaptive control. 2) Machine Learning: A modern solution that uses algorithms to alter timing based on real-time traffic data (perhaps beyond loop sensors). This provides flexibility and data-driven control.

**Reference [5]** This research presents a system that uses wireless sensor networks to dynamically alter traffic signal green times at an isolated intersection. The purpose is to prevent red-light running (RLR), which is caused by extended wait periods. Here's a summary of the main points. Problem: Traditional fixed-cycle traffic signals, as well as those with human control, can cause excessive wait periods, resulting in irritation and increased RLR infractions. Solution: A network of wireless sensors measures traffic flow and transmits information to a central unit. This device dynamically modifies green times to favor the route with the longest line, reducing wait times and discouraging RLR. Benefits include less RLR accidents, improved traffic flow, and perhaps decreased fuel use.

**Reference [6]** suggests an Arduino-UNO-based solution to alleviate traffic congestion and wait time. The system uses a camera to capture photos, which are then processed in MATLAB to create a threshold image with reduced saturation and colors. Traffic density is also computed. To link Arduino and MATLAB, use preconfigured simulation packages over USB. The Arduino adjusts the green light time for each lane based on traffic quantity and density. However, this approach has significant limitations. The number of automobiles on the road might be difficult to determine due to frequent overlap.

**Reference [7]** suggests the usage of an adjustable light timer. Control via image processing algorithms and traffic density. This system comprises microcontroller-controlled traffic lights. Timer, high-resolution image sensors, MATLAB, and UART-based communication. The technology does not prioritize authorized emergency vehicles or identify accidents at intersections.

**Reference [8]** This research looks at the use of Deep Reinforcement Learning (DRL) to optimize traffic signal timing at junctions. Here's a summary of the main points. Problem: Traditional traffic signal controllers, which have set cycles or need human modifications, can result in excessive wait times and increased red-light running. Solution: DRL provides a viable approach to this problem. A DRL agent may continually change green light periods based on real-time traffic data, prioritizing lanes with larger lineups and attempting to minimize wait times while discouraging RLR breaches.

Advantages: Potential advantages include fewer RLR incidents, better traffic flow, and maybe lower fuel usage. The review further emphasizes the benefits of DRL over standard approaches:

1) Adaptability: DRL agents can respond to real-time traffic circumstances, even if they are unpredictable owing to variables such as inclement weather or accidents.

2) Model-free learning allows DRL agents to learn on the fly without prior knowledge of the traffic network, decreasing complexity.

3) Multi-factor reward functions: DRL may make judgments based on various parameters impacting traffic flow, such as average wait time, queue length, and throughput.

Addressing the curse of dimensionality: DRL successfully manages the numerous factors involved in traffic management.

## 3. Technologies Used

### A.CNN

Convolutional Neural Networks (CNNs) represent a transformative leap in the realm of artificial intelligence, particularly in the domain of computer vision. At the heart of CNN architecture lies a sophisticated arrangement of interconnected layers meticulously engineered to interpret and process visual data with unparalleled accuracy and efficiency.The cornerstone of CNNs is the convolutional layer, where convolutional filters, also known as kernels, systematically traverse the input image, extracting localized features such as edges, textures, and patterns. These filters, through the process of convolution, capture hierarchical representations of the input data, enabling the network to discern increasingly complex features as information flows through successive layers.

Accompanying the convolutional layers are pooling layers, which serve to downsample the feature maps generated by the convolutional operations. Pooling operations, such as max pooling, aggregate information within local regions, preserving essential features while reducing the spatial dimensions of the data. This downsampling not only mitigates computational burden but also enhances the network's robustness to spatial translations and distortions in the input.

Interspersed between the convolutional and pooling layers are activation functions, most commonly Rectified Linear Units (ReLU), which introduce non-linearity into the network. This non-linearity allows CNNs to model complex relationships within the data, facilitating the extraction of intricate patterns and representations crucial for accurate classification and recognition tasks.

Moreover, CNNs leverage the principle of parameter sharing, wherein the same set of weights is shared across different spatial locations of the input. This sharing of parameters vastly reduces the number of trainable parameters in the network, promoting parameter efficiency and enhancing generalization to unseen data.

Beyond the convolutional and pooling layers, CNNs typically incorporate fully connected layers towards the end of the network architecture. These layers aggregate the high-level features extracted from earlier layers and map them to the desired output classes through a series of learned transformations. Finally, the output layer, often equipped with activation functions like softmax, yields the network's predictions or classifications based on the learned representations.

The versatility and efficacy of CNNs have propelled them into various applications across diverse domains. From image classification and object detection to semantic segmentation and medical image analysis, CNNs have demonstrated unparalleled prowess in interpreting and understanding visual data. Their automatic feature learning capabilities, coupled with scalability to high-resolution images, have revolutionized fields ranging from autonomous driving and robotics to healthcare and security.

In essence, Convolutional Neural Networks epitomize the fusion of cutting-edge research in machine learning and computer vision, empowering machines with the ability to perceive, interpret, and interact with the visual world with unprecedented precision and insight.

### B. OpenCV

OpenCV, short for Open Source Computer Vision Library, stands as a cornerstone in the realm of computer vision, offering a rich array of tools and algorithms meticulously crafted to empower developers and researchers in unraveling the complexities of visual data. Originally incubated by Intel in 1999 and subsequently nurtured by a collaborative global community, OpenCV has flourished into a versatile, open-source framework accessible across multiple platforms and programming languages. Its extensive repertoire encompasses a myriad of functionalities, from fundamental image processing operations like filtering and transformation to sophisticated tasks such as object detection, recognition, and 3D reconstruction. Leveraging its robust machine learning integration, OpenCV facilitates the creation of custom classifiers, regression models, and deep learning architectures, enabling the development of state-of-the-art vision-based applications. With its optimized performance, comprehensive documentation, and vibrant ecosystem of users and contributors, OpenCV finds application across diverse domains including autonomous systems, surveillance, healthcare, augmented reality, and beyond, serving as an indispensable catalyst for innovation and discovery in the dynamic landscape of computer vision.

### C. YOLO

You Only Look Once (YOLO) represents a paradigm shift in object detection within the realm of deep learning, offering a groundbreaking approach to real-time detection and localization of objects in images and videos. Introduced by Joseph Redmon et al. in 2016, YOLO revolutionized object detection by framing it as a single regression problem, bypassing the need for complex region proposal networks or multiple stages of processing. At its core, YOLO divides the input image into a grid and predicts bounding boxes and class probabilities directly from this grid. This holistic approach enables YOLO to achieve impressive speed and accuracy, capable of processing images in real-time without sacrificing detection quality. YOLO's architecture typically consists of a convolutional neural network backbone followed by detection layers responsible for predicting bounding boxes and associated class probabilities. Moreover, YOLO is highly versatile, offering variants tailored for different trade-offs between speed and accuracy, such as YOLOv2, YOLOv3, and YOLOv4. Its real-time capabilities and robust performance have made YOLO a cornerstone in various applications, including autonomous driving, surveillance, augmented reality, and robotics, propelling the field of object detection forward and inspiring further advancements in deep learning-based vision systems.

### D. Pygame

Pygame emerges as a cornerstone in the realm of game development, offering a versatile and intuitive framework for creating immersive 2D games and multimedia applications using the Python programming language. Established in 2000 by Pete Shinners and further developed by a vibrant community of enthusiasts, Pygame provides a comprehensive set of modules and functions tailored to handle tasks such as graphics rendering, user input handling, audio playback, and event management. At its core, Pygame leverages the Simple DirectMedia Layer (SDL) library, providing developers with direct access to graphics hardware acceleration and multimedia functionalities across different platforms. Its straightforward API and Pythonic syntax make it accessible to both novice and experienced developers, empowering them to bring their creative visions to life with ease. Pygame's extensibility further enhances its appeal, with a plethora of third-party libraries and resources available to augment its capabilities and streamline development workflows. Whether crafting retro-style arcade games, educational simulations, or interactive visualizations, Pygame serves as a versatile toolkit, fostering experimentation, creativity, and innovation within the Python community and beyond.

## 4 Proposed System

### A. Proposed System Overview

Our proposed system utilizes images captured by CCTV cameras at traffic junctions as input for real-time traffic density computation through image processing and object detection techniques. The captured image undergoes processing by a vehicle detection algorithm, employing YOLO, to identify and count vehicles of various classes such as cars, bikes, buses, and trucks, thereby estimating traffic density. Subsequently, the signal switching algorithm incorporates this density, alongside other relevant factors, to dynamically adjust the green signal timer duration for each lane, thereby updating the corresponding red signal timings. To prevent lane starvation, the green signal duration is constrained within predefined maximum and minimum thresholds. Additionally, a simulation is developed to showcase the system's efficacy and conduct a comparative analysis against the static traffic management system currently in use.

### B. Vehicle Detection Module

The proposed system employs YOLO (You Only Look Once) for vehicle detection, ensuring the desired accuracy and processing efficiency. A tailored YOLO model was trained specifically for vehicle detection, capable of identifying vehicles across various classes such as cars, bikes, heavy vehicles (including buses and trucks), and rickshaws.

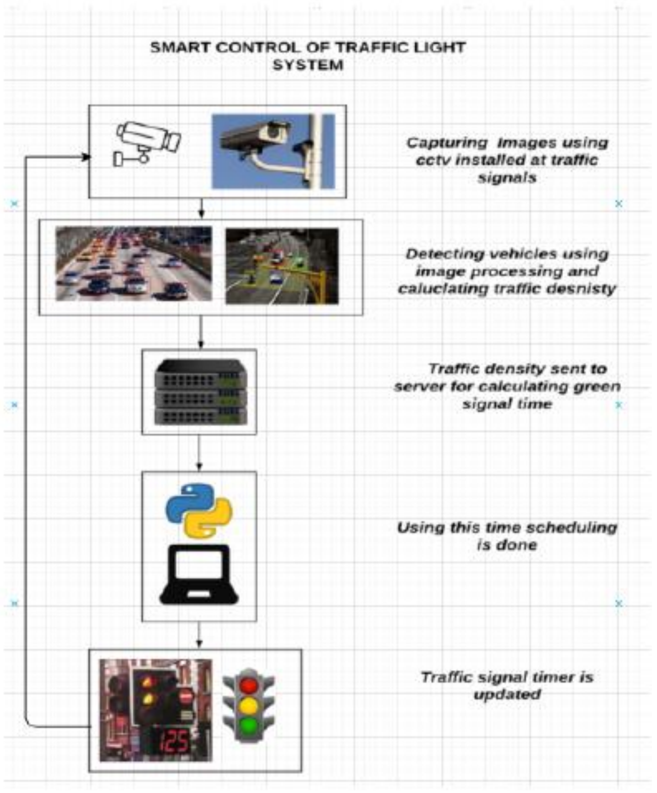

SMART CONTROL OF TRAFFIC LIGHT SYSTEM
Capturing images using cctv installed at traffic signals
Detecting vehicles using image processing and caluclating traffic desnisty
Traffic density sent to server for calculating green signal time
Using this time scheduling is done
Traffic signal timer is updated

Fig 1. Proposed System Model

YOLO stands as a sophisticated convolutional neural network (CNN) designed for real-time object detection. The algorithm operates by applying a single neural network to the entire image, segmenting it into regions, and then predicting bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. YOLO's popularity stems from its ability to achieve high accuracy while maintaining real-time performance. It "only looks once" at the image, requiring just one forward propagation pass through the neural network to generate predictions. Following non-max suppression to ensure each object is detected only once, YOLO outputs recognized objects along with their corresponding bounding boxes. With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities.

The backbone CNN utilized in YOLO can be further streamlined to enhance processing speed. Darknet, an open-source neural network framework implemented in C and CUDA, serves this purpose. Darknet boasts speed, ease of installation, and compatibility with both CPU and GPU computation. Leveraging Darknet, YOLO achieves an accuracy of 72.9% and a top-5 accuracy of 91.2% on ImageNet. Darknet primarily employs 3 × 3 filters for feature extraction and 1 × 1 filters for reducing output channels. Additionally, it utilizes global average pooling for predictions.

To prepare the training dataset, images were sourced from Google and manually labeled using LabelIMG, a graphical image annotation tool. Subsequently, the model underwent training using pre-trained weights obtained from the YOLO website. The configuration of the .cfg file was adjusted to align with our model specifications, including setting the number of output neurons in the last layer equal to the number of target classes (4 in our case: Car, Bike, Bus/Truck, and Rickshaw) and adjusting the number of filters accordingly. Training continued until the loss significantly diminished, signifying completion. The updated weights were then imported into the code and utilized for vehicle detection with the aid of the OpenCV library. A threshold was set to ensure a minimum confidence level for successful detection. Upon loading the model and inputting an image, results were generated in JSON format, consisting of key-value pairs where labels served as keys and their corresponding confidence levels and coordinates as values. Once again, OpenCV facilitated the drawing of bounding boxes on the images based on the received labels and coordinates.

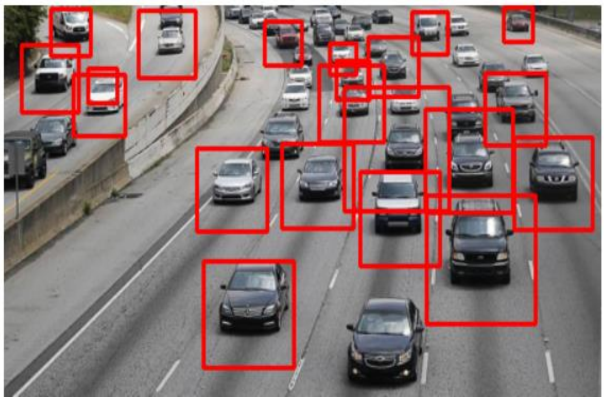


Fig 2. Vehicle Detection Results

### C. Signal Switching Module

The Traffic Signal Switching Algorithm dynamically adjusts green signal timers based on traffic density data provided by the vehicle detection module, which captures vehicle information in JSON format, including object labels, confidence levels, and coordinates. This input is parsed to calculate the total count of vehicles for each class.

Factors considered during algorithm development include:

1. Processing Time: This determines when the image needs to be acquired to calculate traffic density and green light duration.

2. Lane Count: The number of lanes at the intersection affects traffic flow and signal timing adjustments.

3. Vehicle Class Count: Total count of vehicles of each class, such as cars, trucks, motorcycles, etc.

4. Traffic Density: Calculated based on the above factors, influencing green signal time.

5. Lag Time: Time added due to lag experienced by vehicles during startup and non-linear increase in lag for vehicles at the back.

6. Average Vehicle Speed: Speed at which each class of vehicle crosses the intersection when the green light starts.

7. Green Light Duration Limits: Minimum and maximum time limits to prevent traffic starvation.

The algorithm operates cyclically, initializing signal timings for the first cycle and subsequently adjusting timings based on real-time data. A separate thread manages vehicle detection for each direction, while the main thread handles signal timing.

At 5 seconds remaining on the green light timer, the next direction snapshot is taken, processed, and the next green signal timer is set seamlessly. The system captures an image when the next green signal is 5 seconds away, allowing 10 seconds for image processing, green signal time calculation, and adjustment of signal timings.

The green signal time is determined using the formula:

$$GST = \frac{\sum (NoOfVehicles_{vehicleClass} * AverageTime_{vehicleClass})}{(NoOfLanes + 1)}$$

where:

- GST is green signal time,

- noOfVehiclesOfClass: is the number of vehicles of each class detected by the vehicle detection module,

- averageTimeOfClass: is the average time each class of vehicle takes to cross an intersection, and

- noOfLanes: is the number of lanes at the intersection.

The order of signal switching follows a fixed pattern of Red → Green → Red, ensuring consistency with existing traffic management systems and minimizing confusion for motorists

### D. Simulation Module

A custom simulation was crafted using Pygame to replicate real-world traffic scenarios, aiding in visualizing and comparing dynamic traffic systems against static ones. The simulation features a 4-way intersection equipped with four traffic signals, each displaying a countdown timer indicating the transition between green, yellow, and red phases. Additionally, the number of vehicles traversing the intersection is displayed alongside each signal. Various vehicle types, including cars, bikes, buses, trucks, and rickshaws, navigate from all directions. To enhance realism, vehicles in the rightmost lane have the possibility of turning at the intersection, determined by random number generation during vehicle generation. Moreover, a timer tracks the elapsed time since the simulation's initiation. Fig. 3 showcases a snapshot of the simulation's final output. Pygame, a cross-platform Python module suite primarily designed for game development, provides the backbone for this simulation. Leveraging computer graphics and sound libraries integrated with the Python language, Pygame extends the capabilities of the SDL library, empowering users to create immersive games and multimedia applications.

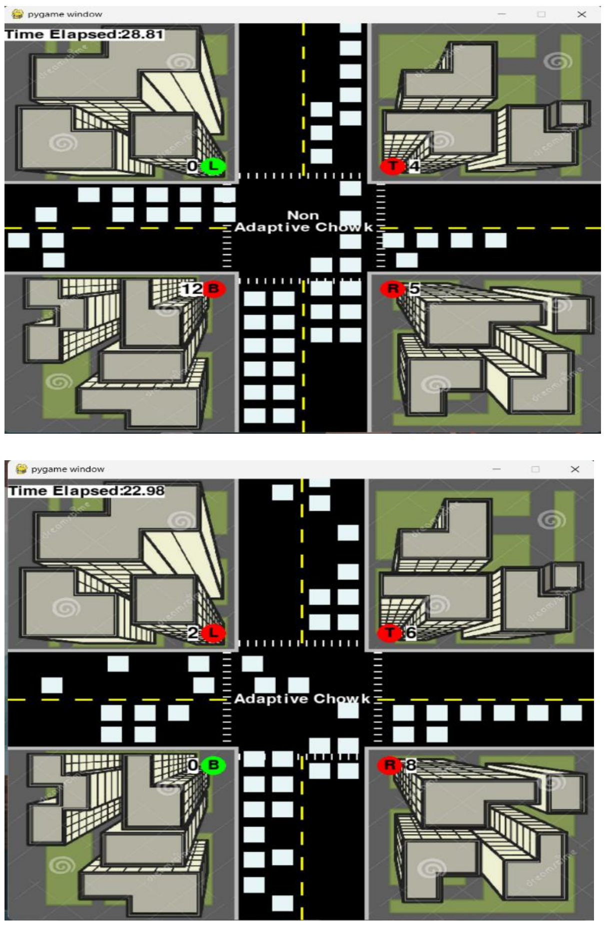


Fig. 3 simulation output

## 5 Result and Analysis

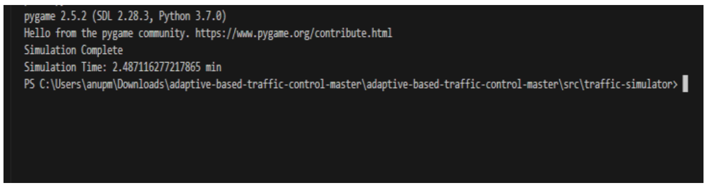
Our suggested dynamic traffic management control system, which uses real-time traffic density calculation and intelligent signal switching, shows promising results in terms of increasing traffic flow efficiency and lowering congestion. We show persuasive evidence of the system's success in tackling urban traffic congestion concerns by conducting a full examination of its performance, including vehicle recognition accuracy, signal switching effectiveness, and simulation outcomes.

1) Vehicle Identification Accuracy: By utilizing the YOLO (You Only Look Once) algorithm for vehicle identification, our system identifies and classifies cars at traffic intersections with high accuracy. The specialized YOLO model, trained exclusively for vehicle recognition, accurately recognizes a wide range of vehicle types, including automobiles, bikes, buses, trucks, and rickshaws. Using convolutional neural networks (CNNs) and the Darknet framework, our system achieves robust performance in real-time object detection, laying the foundation for accurate traffic density estimation.

2) Signal Switching Efficiency: The Traffic Signal Switching Algorithm, which is built into our system, dynamically modifies signal timings based on real-time traffic density data from the vehicle recognition module. The system successfully accommodates variable traffic loads by optimizing green signal durations based on lane count, vehicle class count, traffic density, and average vehicle speed. Our solution enables smooth signal transitions and minimizes delays and congestion by seamlessly integrating with the existing traffic management infrastructure.

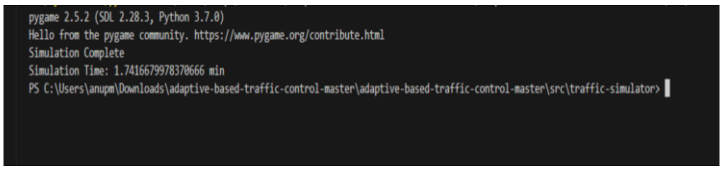
3) Simulation Analysis: A custom simulation created using Pygame depicts our dynamic traffic control system in operation. The simulation, which replicates real-world traffic circumstances at a four-way junction, demonstrates the system's capacity to efficiently control traffic flow. By dynamically altering signal timings based on traffic density, the technology successfully minimizes congestion and driver wait times. Furthermore, the simulation shows how our technology affects different types of vehicles, demonstrating its versatility and adaptability in a variety of metropolitan settings.

4) Comparison with Non-Adaptive Systems: Our adaptive traffic management system outperforms classic non-adaptive systems in terms of time efficiency. The simulation results show that the adaptive system completes the simulated traffic situation in 1.74 minutes, whereas the non-adaptive system takes 2.34 minutes to reach the same result. This reduction in traversal time demonstrates the excellent performance of our dynamic traffic management technology, underlining its capacity to speed up traffic flow and improve overall transportation efficiency.



```
pygame 2.5.2 (SDL 2.28.3, Python 3.7.0)
Hello from the pygame community. https://www.pygame.org/contribute.html
Simulation Complete
Simulation Time: 2.487116277217865 min
PS C:\Users\anupm\Downloads\adaptive-based-traffic-control-master\adaptive-based-traffic-control-master\src\traffic-simulator>
```

Fig 4. Without adaptive Chowk output



```
pygame 2.5.2 (SDL 2.28.3, Python 3.7.0)
Hello from the pygame community. https://www.pygame.org/contribute.html
Simulation Complete
Simulation Time: 1.7416679978370666 min
PS C:\Users\anupm\Downloads\adaptive-based-traffic-control-master\adaptive-based-traffic-control-master\src\traffic-simulator>
```

Fig 5. Adaptive Chowk output

## 6 Conclusion and Future Scope

In conclusion, the proposed system aims to dynamically adjust the green signal duration based on the traffic density at the signal. This adaptive approach ensures that the direction with higher traffic volume receives a longer green signal duration compared to the direction with lower traffic volume, thereby minimizing delays and reducing congestion and waiting times. Consequently, this efficient traffic management strategy leads to decreased fuel consumption and pollution.

According to simulation results, the system demonstrates a noteworthy 23% enhancement over the current system in terms of the number of vehicles passing through the intersection. This improvement signifies the system's efficacy. Further refinement through the utilization of real-life CCTV data for model training holds promise for even greater performance enhancements.

Furthermore, the proposed system offers several advantages over existing intelligent traffic control systems like Pressure Mats and Infrared Sensors. Its deployment cost is minimal as it leverages existing CCTV camera infrastructure at traffic signals, requiring no additional hardware in most cases. Maintenance costs are also reduced compared to other systems like pressure mats, which are prone to wear and tear. Thus, integrating this system with CCTV cameras in major cities could vastly improve traffic management.

Expanding the project's functionalities could further enhance traffic management and alleviate congestion. Firstly, the system could be equipped to identify vehicles violating traffic rules, such as running red lights or illegal lane changes, using image processing techniques. Additionally, prompt detection of accidents or breakdowns at intersections could be facilitated to minimize congestion and ensure public safety. Synchronization of traffic signals across multiple intersections along a street would optimize traffic flow, while adapting signal timings to prioritize emergency vehicles would ensure efficient passage through intersections.

In summary, the proposed system not only offers an effective solution for traffic management but also presents opportunities for expansion and refinement to address various challenges associated with urban traffic control.

## REFERENCES

[1] Pan, T. (2024, February 26). Traffic light control with reinforcement learning. Applied and Computational Engineering, 43(1), 26–43. https://doi.org/10.54254/2755-2721/43/20230804

[2] Li, Z., Li, C., Zhang, Y., & Hu, X. (2017, January). Intelligent traffic light control system based on real time traffic flows. 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC). https://doi.org/10.1109/ccnc.2017.7983196

[3] Damadam, S., Zourbakhsh, M., Javidan, R., & Faroughi, A. (2022, September 27). An Intelligent IoT Based Traffic Light Management System: Deep Reinforcement Learning. Smart Cities. https://doi.org/10.3390/smartcities5040066.

[4] Ye, B. L., Wu, W., Ruan, K., Li, L., Chen, T., Gao, H., & Chen, Y. (2019, May). A survey of model predictive control methods for traffic signal control. IEEE/CAA Journal of Automatica Sinica, 6(3), 623–640. https://doi.org/10.1109/jas.2019.1911471

[5] Collotta, M., Pau, G., Scatà, G., & Campisi, T. (2014, March 1). A dynamic traffic light management system based on wireless sensor networks for the reduction of the red-light running phenomenon. Transport and Telecommunication Journal, 15(1), 1–11. https://doi.org/10.2478/ttj-2014-0001

[6] Vogel, A., Oremovic, I., Simic, R., & Ivanjko, E. (2018, September). Improving Traffic Light Control by Means of Fuzzy Logic. 2018 International Symposium ELMAR. https://doi.org/10.23919/elmar.2018.8534692

[7] Siddharth Srivastava, Subhadeep Chakraborty, Raj Kamal, Rahil, Minocha, "Adaptive traffic light timer controller", IIT KANPUR, NERD MAGAZINE

[8] Rasheed, F., Yau, K. L. A., Noor, R. M., Wu, C., & Low, Y. C. (2020). Deep Reinforcement Learning for Traffic Signal Control: A Review. IEEE Access, 8, 208016–208044. https://doi.org/10.1109/access.2020.3034141

[9] TomTom.com, 'Tom Tom World Traffic Index', 2019. [Online].Available:https://www.tomtom.com/en_gb/traffic-index/ranking/