ISSN 2277-2685
IJESR/June. 2025/ Vol-15/Issue-3s/406-414
Mudigiri Sankalpa *et. al.*, / International Journal of Engineering & Science Research

Smart Traffic Control System Using Image AI

P Mounika, Mudigiri Sankalpa, Dandu Sathvika Padmavathi, Kancharakuntla Sreeja
[1]Associate Professor, Department Of Cse, Bhoj Reddy Engineering College For Women, India.
[2,3,4]B. Tech Students, Department Of Cse, Bhoj Reddy Engineering College For Women, India.

ABSTRACT

Learning-based traffic control algorithms have recently been explored as an alternative to existing traffic control logics. The reinforcement learning (RL) algorithm is being spotlighted in the field of adaptive traffic signal control. However, no report has described the implementation of an RL-based algorithm in an actual intersection. Most previous RL studies adopted conventional traffic parameters, such as delays and queue lengths to represent a traffic state, which cannot be exactly measured on-site in real time. Furthermore, the traffic parameters cannot fully account for the complexity of an actual traffic state. The present study suggests a novel artificial intelligence that uses only video images of an intersection to represent its traffic state rather than using handcrafted features. In simulation experiments using a real intersection, consecutive aerial video frames fully addressed the traffic state of an independent four-legged intersection, and an image-based RL model outperformed both the actual operation of fixed signals and a fully actuated operation.

1. INTRODUCTION

Smart Traffic Control Systems range from basic queue-based methods to advanced algorithms like optimal control theory. Recently, reinforcement learning (RL) has shown promise in traffic management due to its ability to optimize real-time control without relying on pre-set training data. RL agents can manage signals for single or multiple intersections, continuously improving control decisions. Simulations have shown RL's superiority over traditional methods, suggesting it may dominate future traffic control.

However, RL-based control faces two key challenges in real-world applications. First, accurately recognizing traffic states is crucial for effective RL decision-making. Conventional parameters like delays and queue lengths are difficult to measure in real-time with current surveillance systems. The adoption of vehicle-to-junction communication technology, necessary for accurate data collection, is still in progress. As a result, most RL models remain limited to simulations.

Second, traditional traffic parameters may not fully capture the complexity of traffic states. This study proposes a solution by integrating RL with convolutional neural networks (CNNs) to analyse aerial images of intersections, offering a more comprehensive method for recognizing traffic conditions and optimizing signal control. Simulation results demonstrate the feasibility of this approach, highlighting its potential for real-world applications and suggesting the need for further research to implement RL-based traffic control in practice.

Existing System

In the current traffic management system, controlling traffic flow often requires manual intervention. Traffic police or operators assess which side of an intersection has the heaviest traffic and then adjust the signal timing to provide a green light for a longer duration on that side. While this approach may work during peak traffic hours, it has significant limitations, especially during off-peak times like late at night or early in the morning. Manual traffic

management is labor-intensive, time-consuming, and prone to human error, which can lead to inefficiencies and delays.

Additionally, relying on manual adjustments makes it challenging to respond to dynamic traffic patterns in real-time. Factors such as unexpected surges in traffic due to accidents or events cannot always be handled promptly. During late hours, when operators are not consistently present, the system often relies on fixed signal timings, which fail to adapt to real-time traffic conditions.

To address these limitations, artificial intelligence (AI) and automated traffic control systems have been introduced. By utilizing AI, traffic signals can be controlled automatically based on real-time data, ensuring smoother traffic flow without the need for constant human intervention. These systems are particularly effective during times when manual control is impractical, significantly improving overall traffic efficiency.

Proposed System

To overcome the limitations of the existing traffic system, an object detection-based approach is proposed to automate traffic management. This system uses input images or video footage from traffic cameras to monitor traffic conditions. Object detection algorithms analyze the images to detect and count vehicles, calculating the traffic density as a percentage.

The calculated percentage is then compared to a predefined threshold value. Based on this comparison, the system dynamically adjusts signal timings. Intersections with heavier traffic receive longer green signals, while areas with lighter traffic get shorter green signals.

This automated method eliminates the need for manual intervention, enabling real-time traffic monitoring and efficient signal control. It significantly reduces delays, congestion, and fuel consumption. By adapting to current traffic conditions, this system not only improves traffic flow but also decreases environmental impacts, offering a smarter and more sustainable solution for managing traffic effectively.

2-REQUIREMENT ANALYSIS

Functional Requirements

Functional requirements define the core operations that the system must perform. Functional requirements specify the actions or tasks a system must perform to meet user needs and achieve its objectives. These requirements outline what the system should do, including specific features, functionalities, and interactions. They describe the system's behavior, such as processing data, generating reports, executing calculations, or providing user interfaces for specific operations. Functional requirements are directly tied to the purpose of the system and ensure that it delivers the desired outcomes, aligning with user expectations and project goals. These are derived from the objectives of the traffic control system and include all necessary features to ensure it functions as intended.

Admin Module

In this project, the Admin is the sole module and represents the entire system. All functionalities are handled automatically by the system, eliminating the need for any manual supervision.

This module allows an administrator to manage the adaptive traffic signal control system. It includes the following core functionalities:

Non-Functional Requirements

Non-functional requirements define how a system operates rather than what it does. They focus on the quality attributes and performance of the system, such as usability, reliability, security, and scalability.

These requirements ensure the system is intuitive to use, consistently available, and capable of handling varying workloads efficiently. Non-functional requirements also address aspects like compatibility with different platforms, ease of maintenance, and speed of execution. They are crucial for ensuring that the system not only fulfills its intended functions but does so in a manner that is effective, secure, and user-friendly.

Non-functional requirements describe the quality attributes of the system. These ensure that the system performs efficiently, securely, and is user-friendly.

Portability

- The system should run on various operating systems like Windows, Linux, and macOS.
- Minimal changes should be required for deployment on different platforms.
- All dependencies should be cross-platform compatible.

Reliability

- The system must provide consistent and stable output under all traffic conditions.
- It should be fault-tolerant and capable of recovering from failures.
- System performance should not degrade over time.

Usability

- The user interface should be simple and intuitive.
- Non-technical users should be able to operate the system easily.
- Visuals and controls must be clearly labeled and user-friendly.

Hardware Requirements

Hardware Requirements are the most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware.

- Processor : Intel i3, i5
- RAM : 4 GB
- Hard Disk : 100GB
- Graphics : GPU support for deep learning model training

Software Requirements

The software requirements document is the specification of the system. It should include both definition and a specification of the requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team's progress throughout the development activity.

- Operating System : Windows 11
- Programming Language : Python 3.6
- IDE : Jupyter Notebook
- UML Tool : StarUML
- DFD Tool : DFD Drawer
- Libraries : OpenCV, NumPy, TensorFlow

3-DESIGN

Architecture

Architecture refers to the high-level structure of a system, defining its components, their relationships, and how they work together to achieve specific goals. It serves as a blueprint for design and development, guiding the implementation process while ensuring alignment with functional and non-functional requirements.

Key Elements of Architecture:

1. System Components:
   Describes the distinct elements or modules that perform specific functions within the system.
2. Interconnections:
   Specifies how components interact, including communication protocols, data flow, and interfaces.
3. Data Architecture:
   Defines how data is collected, stored, processed, and accessed to ensure consistency and reliability.
4. Technology Stack:
   Identifies the tools, frameworks, and platforms used for development and deployment.
5. Design Patterns:
   Incorporates proven, reusable solutions to address recurring design challenges effectively.
6. Non-Functional Considerations:
   Ensures the architecture accounts for performance, scalability, security, reliability, and maintainability.
7. Deployment Strategy:
   Outlines how the system is implemented in the infrastructure, including configurations and environments.
8. Integration Points:
   Details how the system interfaces with external systems, services, or APIs.

Software Architecture

A well-designed architecture is essential for creating systems that are efficient, scalable, maintainable, and capable of adapting to changing requirements. It provides a framework for collaboration among stakeholders, ensuring the system meets its intended purpose.

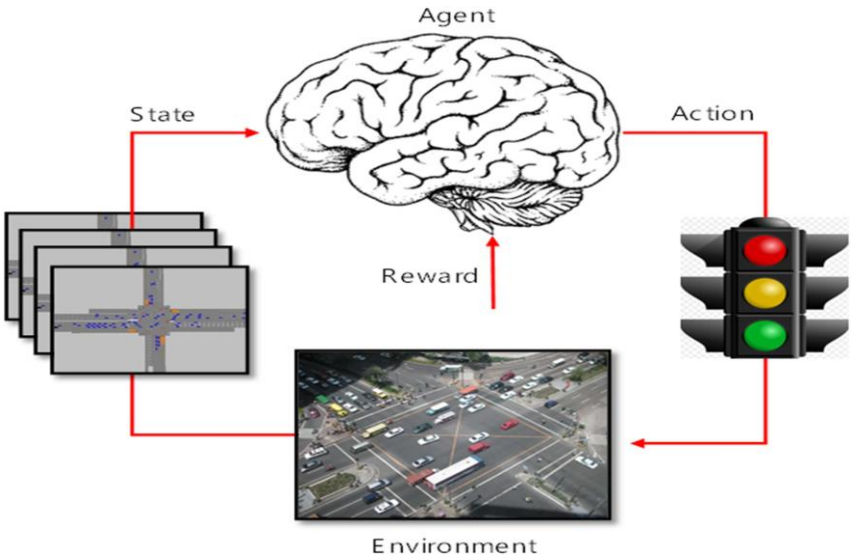

Fig.3.1.Software Architecture

In software architecture a Reinforcement Learning (RL) based Smart Traffic Control System, where an AI agent (depicted as a brain) learns to manage traffic lights by interacting with its environment (the road intersection). The environment includes live traffic conditions captured via surveillance cameras. The agent observes the state of traffic—vehicle count, lane density, and signal timings—and uses this to make decisions.

The state represents traffic input data, possibly as preprocessed images or numerical representations of vehicle positions and counts from multiple lanes. The agent processes this state and takes an action—changing the traffic light (red, yellow, or green) for a particular lane. This action directly influences the traffic flow and affects how vehicles move through the intersection.

After performing an action, the system receives a reward based on its effectiveness—such as reduced waiting time, fewer traffic jams, or balanced lane usage. This reward helps the agent learn better strategies over time. Through many such interactions, the agent gradually optimizes its control policy, creating a dynamic and intelligent traffic light system that adapts to real-time road conditions.

Technical Architecture

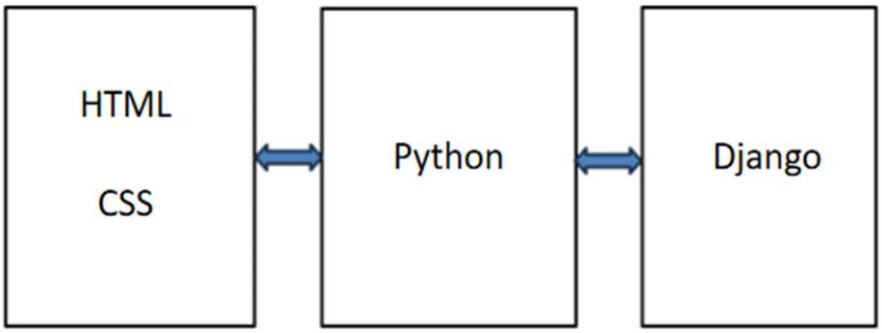

Fig.3.2.Technical Architecture

The technical architecture is simplified web technology stack for implementing the Smart Traffic

Controlling System. The left block represents HTML and CSS, which are responsible for designing the frontend UI—the part of the system that users interact with. Through this interface, users can monitor live traffic visuals, view the vehicle count per lane, and check signal status. It can also include controls for manual override and system configuration.

The middle block represents Python, which acts as the core programming language used for the backend logic. Python handles communication between the frontend and the backend logic, processes data (like vehicle count received from the YOLO model), and controls the flow of information. In this project, Python also integrates with image processing libraries
(like OpenCV) and AI models to analyze traffic camera inputs and make smart decisions based on the data.

The final block represents Django, a powerful Python-based web framework. Django helps in managing the web server, routing requests, connecting to databases, and serving HTML templates dynamically. It also manages API calls that fetch real-time traffic data or trigger signal changes. Together, Django and Python ensure that the AI decisions made by the system are effectively integrated into a working web-based traffic control interface.

4-IMPLEMENTATION

Python

Python is a powerful, high-level, interpreted programming language that has gained immense popularity due to its simplicity and flexibility. Created by Guido van Rossum and first released in 1991, Python was designed with an emphasis on code readability and a syntax that allows developers to express concepts in fewer lines of code compared to other programming languages such as Java or C++. It follows multiple programming paradigms, including procedural, object-oriented, and functional programming.

As an open-source language, Python has a vibrant community and an extensive ecosystem of libraries and frameworks. Whether it's web development, machine learning, data analysis, desktop applications, or scripting, Python offers tools and packages that make development easier and more efficient. Python's clean and readable syntax makes it an excellent choice for both beginner programmers and experienced developers who are working on complex applications.

Python is platform-independent, which means that Python programs can run on different operating systems like Windows, Linux, and macOS without modification. Furthermore, Python is dynamically typed, meaning you don't need to specify variable types — Python automatically infers the data type at runtime. These features, among many others, make Python an ideal language for rapid application development and a staple in both academic and industrial environments.

Key Features of Python

1. **Easy-to-Read and Clean Syntax**

- Purpose: Enhances readability and reduces code complexity.
- Usage: Python uses indentation (whitespace) to define blocks instead of curly braces.

2. **Dynamically Typed Language**

- Purpose: Avoids the need for variable type declaration.
- Usage: Python identifies the type automatically at runtime.

3. **Extensive Standard Library**

- Purpose: Provides built-in modules for common programming tasks.
- Usage: Easily perform operations like file handling, date/time management, etc.

4. Interpreted and Interactive

- Purpose: Supports quick testing and real-time debugging.
- Usage: Use Python shell to test code instantly.

5. Cross-Platform Compatibility

- Purpose: Run the same code on multiple OS platforms.
- Usage: Python scripts run without modification on Windows, Linux, and macOS.

6. Object-Oriented Programming (OOP)

- Purpose: Promotes code reuse and encapsulation.
- Usage: Supports classes, objects, and inheritance.

Advantages of Python

1. Beginner-Friendly Language

   Python's simple syntax, close to English, makes it an ideal first language for new programmers. You can start writing functional programs with minimal learning curve, which encourages experimentation and fast learning.

2. Vast Standard Library

   Python comes with a comprehensive standard library that includes modules for file I/O, regular expressions, threading, networking, and even web services. This reduces development time and avoids the need to reinvent the wheel.

3. Large Community Support

   Python boasts one of the largest programming communities. This means a wealth of tutorials, forums, documentation, and third-party tools are available, helping both beginners and professionals solve problems faster.

5-SCREENSHOTS

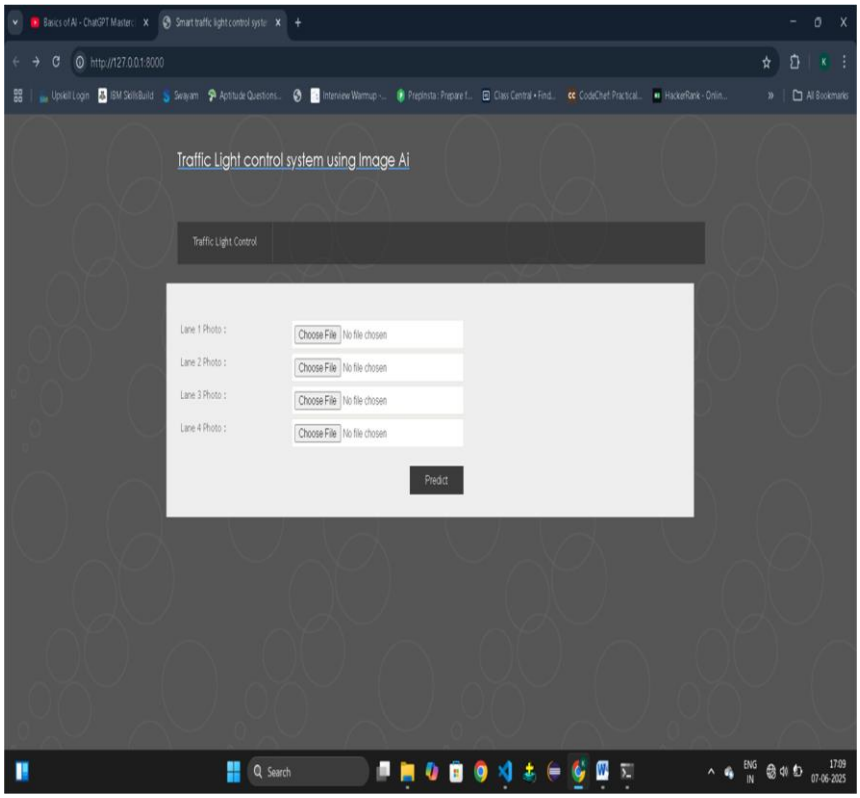

Fig.1. Index page

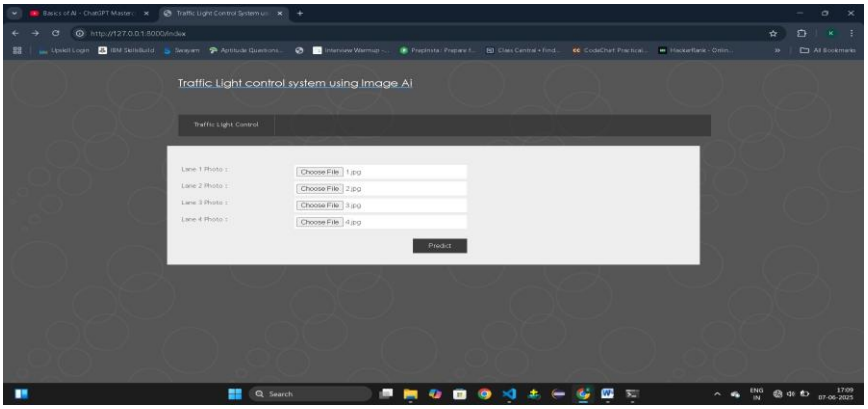

Fig.2. Uploading the Images

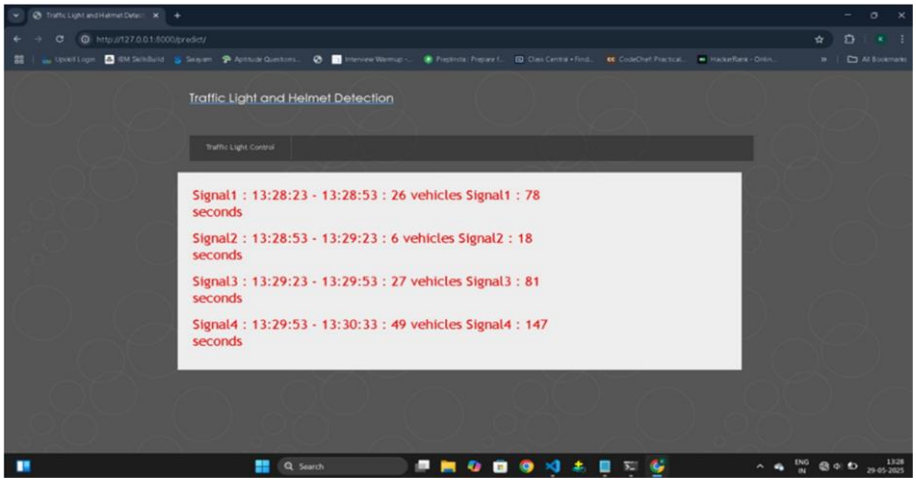

Fig.3. Result page

6-CONCLUSION

This study introduces a pioneering image-based Reinforcement Learning (RL) algorithm for traffic signal control, offering an alternative to traditional methods that rely on traffic detectors. Unlike previous models that depend on predefined traffic parameters, this approach uses real-time image data to represent the traffic state, making it more adaptive and scalable. The algorithm successfully reduced average vehicle delay by more than 23% and outperformed both actual and fully actuated signal operations in all three evaluated performance measures.

One of the key performance indicators, the WAVE (Weighted Average Vehicle Excess), also showed noticeable improvement. The proposed model recorded a mean WAVE reduction of 21% compared to actual operations and 12% compared to fully actuated systems. These improvements validate the potential of image-based RL in enhancing traffic flow and minimizing congestion without the need for expensive detector infrastructure.

However, the model's performance stability remains a concern. Although it achieved overall superior results, the delay times fluctuated more than those observed in reference models. This inconsistency highlights the importance of identifying and tuning critical hyper- parameters. Additionally, the model's current application is limited to a single independent intersection, suggesting the need for scalability and broader testing in more complex traffic networks.

REFERENCES

[1] Boureau, Y.L., Ponce, J. and LeCun, Y. (2023) A Theoretical Analysis of Feature Pooling in Visual Recognition. In International Conference on Machine Learning

[2] Chung, J. and Sohn, K. (2023) Image-based learning to measure traffic density using a deep convolutional neural network (CNN), IEEE Transactions on Intelligent Transport Systems, DOI:10.1109/TITS.2017.2732029

[3] Abdulhai, B., Kattan, L.: (2022) Reinforcement learning: Introduction to theory and potential for transport applications. Canadian Journal of Civil Engineering 30(6), 981–991.

[4] Abdulhai, B., Pringle, R., Karakoulas, G.: (2022) Reinforcement learning for true adaptive traffic signal control. Journal of Transportation Engineering 129(3), 278–285.

[5] Arel, I., Liu, C., Urbanik, T., Kohls, A.: (2021) Reinforcement learning-based multi-agent system for network traffic signal control. Intelligent Transport Systems, IET 4(2), 128–135

[6] Baird., L. (2021) Residual algorithms: Reinforcement learning with function approximation.
ICML, pages 30–37

[7] Abdoos, M., Mozayani, N., Bazzan, A.: (2020) Hierarchical control of traffic signals using q-learning with tile coding. Applied Intelligence 40(2), 201–213.

[8] Dauphin, Y. N., de Vries, H., Chung, J., &Bengio, Y. (2020). RMSProp and equilibrated adaptive learning rates for non-convex optimization. arXiv preprint arXiv:1502.04390.

[9] Corazza, M., &Sangalli, A. (2020). Q-Learning and SARSA: a comparison between two intelligent stochastic control approaches for financial trading. University Ca'Foscari of Venice, Dept. of Economics Research Paper Series No, 15.

[10] Ciresan, D., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J. (2019) Flexible, High Performance Convolutional Neural Networks for Image