Smart City Traffic Light Control Using Reinforcement Learning

Written by Sahitya Biswas, Sayan Ghosh, Pritam Koyal

Ramakrishna Mission Vivekananda Educational and Research Institute
Research institution in Howrah, West Bengal

Abstract

Managing traffic efficiently at busy city intersections is a challenging task, especially when traffic conditions change throughout the day. Most traditional traffic signal systems operate on fixed timers, which often leads to unnecessary waiting and congestion. This project proposes a smart traffic light control system based on reinforcement learning that can adapt its decisions according to real-time traffic conditions at a four-way intersection.

A lane-wise Q-learning approach is used, where the system observes the number of vehicles waiting in each direction and decides which signal should turn green. The entire system is implemented using a custom PyGame simulation that includes realistic vehicle movement, proper stop-line behavior, and emergency vehicle priority. The model learns continuously while the simulation is running, without relying on predefined traffic patterns. The results show a noticeable reduction in vehicle queues and waiting time, demonstrating that reinforcement learning can be an effective and practical solution for intelligent traffic management in smart cities.

Introduction

Urban traffic congestion has become a serious issue in modern smart cities. Rapid urban growth, increasing population, and a rising number of vehicles have placed heavy pressure on existing road networks. Most traditional traffic signal systems still rely on fixed timers or basic rule-based control, which makes them ineffective in handling constantly changing traffic conditions. As a result, vehicles often face long waiting times, large queues, and inefficient traffic movement, particularly during peak hours.

The problem becomes even more critical when emergency vehicles such as ambulances, fire engines, or police vehicles are involved. Delays at intersections can lead to severe consequences, including loss of life or damage to property. This highlights the need for intelligent traffic control systems that can adapt in real time while also ensuring safety and priority for emergency vehicles.

This project focuses on adaptive traffic signal control at a four-way urban intersection using reinforcement learning (RL). Reinforcement learning allows a system to learn optimal decision-making through continuous interaction with its environment. In this work, a Q-learning agent is trained online to control traffic lights by observing lane-wise vehicle congestion and selecting the most appropriate direction for the green signal.

The complete system is implemented using a fully custom PyGame-based simulation. This setup enables realistic vehicle movement, proper lane-wise queue management, emergency vehicle handling, and strict enforcement of stop-line rules. In addition, real-time visualization and performance statistics are provided to monitor traffic flow and learning behavior throughout the simulation.

Objectives of the Project:

To present a clear visual simulation along with meaningful traffic performance statistics.

- To develop a lane-wise adaptive traffic signal controller using reinforcement learning.
- To reduce overall vehicle congestion and average waiting time at the intersection.
- To provide immediate and uninterrupted priority for emergency vehicles.
- To ensure vehicles do not stop inside the intersection and block traffic flow.
- To present a clear visual simulation along with meaningful traffic performance statistics.

Problem Formulation and Representation

Problem Domain

The problem addressed in this project belongs to the domain of *urban traffic signal control*, with particular focus on an isolated four-way road intersection. Such intersections are prevalent in urban environments and are major contributors to traffic congestion due to uneven traffic demand, stochastic vehicle arrivals, and the presence of priority vehicles such as ambulances and fire engines.

The intersection considered in this work consists of four incoming approaches: North, South, East, and West. Vehicles arrive independently at each approach and move toward the center of the intersection. At any given time, traffic signals determine which directions are permitted to proceed, while vehicles in other directions are required to stop and wait. Unlike conventional fixed-time or pre-timed signal systems, the traffic control strategy developed in this project

dynamically adapts signal phases based on real-time traffic conditions.

The environment is inherently dynamic and stochastic. Vehicle arrivals are random, queue lengths evolve continuously, and emergency vehicles may appear unexpectedly and require immediate priority. These characteristics make it difficult to design an optimal control strategy using static or rule-based methods.

From an artificial intelligence perspective, this problem naturally fits the framework of sequential decision-making under uncertainty. Decisions made at a given time step directly influence future traffic states. Therefore, the controller must balance short-term improvements, such as clearing a specific queue, with long-term performance objectives, such as preventing starvation of other directions. Reinforcement learning provides a suitable and effective approach for addressing this problem.

State, Actions, and Constraints

State Representation The state of the traffic environment is represented by a compact numerical vector capturing the current congestion level at the intersection. Formally, the state is defined as:

$$s_t = [q_N(t), q_S(t), q_E(t), q_W(t)] \tag{1}$$

where $q_N(t)$, $q_S(t)$, $q_E(t)$, and $q_W(t)$ denote the number of vehicles currently present in the North, South, East, and West lanes, respectively.

This representation is intentionally simple and interpretable. Rather than modeling individual vehicle positions or velocities, the system focuses on aggregate queue lengths, which are the most relevant indicators for traffic signal control. Despite its simplicity, this state encoding provides sufficient information for the learning agent to identify congestion imbalances and adapt signal timing accordingly.

Action Space The action space consists of a finite set of predefined traffic signal phases. At each decision step, the agent selects one of the following six actions:

1. Allow traffic from the North direction only
2. Allow traffic from the South direction only
3. Allow traffic from the East direction only
4. Allow traffic from the West direction only
5. Allow traffic from North and South simultaneously
6. Allow traffic from East and West simultaneously

These actions correspond to realistic signal configurations commonly used in real-world intersections. Including both single-direction and paired-direction phases provides the agent with sufficient flexibility to handle both localized congestion and balanced opposing traffic flows. The action space is discrete and relatively small, making it well suited for value-based reinforcement learning algorithms such as Deep Q-Learning.

Constraints and Operational Rules To ensure realistic and safe traffic behavior, the simulation environment enforces the following constraints:

- Vehicles must stop at designated stop lines when the signal is red.
- A minimum gap between vehicles is maintained to prevent collisions.
- Only vehicles facing a green signal are permitted to move forward.
- Emergency vehicles are granted priority and may proceed regardless of the current signal phase.
- Once a vehicle enters the intersection, it continues moving until it has completely crossed.

These constraints are hard-coded into the simulation and define the physical and logical boundaries within which the learning agent operates.

Goal Definition The traffic signal control problem does not have a terminal goal state. Instead, it is formulated as a continuous optimization problem. The objective of the agent is to:

- Minimize the total queue length across all directions
- Reduce cumulative vehicle waiting time
- Learn a traffic signal control policy that performs well over extended periods

Performance is evaluated using long-term metrics rather than a single terminal outcome.

Data and Environment

Reinforcement Learning Environment

A custom-built simulation environment was developed using the PyGame library. Rather than relying on a standard OpenAI Gym environment, a bespoke simulator was implemented to allow full control over traffic dynamics, visualization, and reward design.

The environment visually represents roads, lanes, vehicles, stop lines, and traffic signals. Vehicles are generated randomly and follow deterministic motion rules, while traffic signal decisions are controlled by the reinforcement learning agent.

Observation Space The observation space is continuous and consists of a four-dimensional vector representing queue lengths in each direction. This observation is updated at each decision interval and serves as the input to the deep neural network.

Action Space The action space is discrete and contains six possible traffic signal phases, as defined earlier. Each action corresponds to enabling a specific set of directions to move through the intersection.

Reward Structure The reward function is designed to encourage the agent to reduce congestion and waiting time. It is defined as:

$$R_t = \frac{2 \times \Delta Q_t + 1 \times \Delta W_t}{\max(1, N_t)} \tag{2}$$

where ΔQ_t represents the reduction in total queue length, ΔW_t represents the reduction in cumulative waiting time,

and N_t denotes the number of vehicles currently present in the system.

This reward formulation penalizes actions that increase congestion while rewarding decisions that efficiently clear queues. Normalization by the number of vehicles improves learning stability and prevents excessively large reward values.

Episode Structure The simulation timeline is divided into fixed-length segments referred to as *days*. Each day consists of a predefined number of time steps. At the end of each day, average reward and waiting time statistics are recorded for evaluation.

The environment does not terminate at the end of a day. Learning continues across multiple days, allowing the agent to accumulate experience and progressively refine its policy.

Methodology

Algorithm Used: Deep Q-Learning

This project employs Deep Q-Learning (DQN) to solve the traffic signal control problem. DQN combines classical Q-learning with deep neural networks to approximate the optimal action-value function $Q(s, a)$.

The neural network takes the traffic state as input and outputs a Q-value for each possible signal phase. The implementation includes the following key components:

- A policy network for action selection
- A target network to stabilize training
- An experience replay buffer to store past transitions
- An ϵ-greedy exploration strategy to balance exploration and exploitation

Justification for Algorithm Choice

Deep Q-Learning is well suited to this problem due to the low-dimensional continuous state space and discrete action space. The use of neural networks enables generalization across unseen traffic conditions and allows the agent to adapt to dynamic congestion patterns.

Compared to fixed-time controllers, rule-based systems, and tabular Q-learning approaches, DQN provides improved scalability, robustness, and learning stability in complex traffic environments.

Mathematical / Formal Description

The adaptive traffic signal control problem is modeled as a Markov Decision Process (MDP), defined by the tuple

$$\mathcal{M} = (S, A, T, R, \gamma) \tag{3}$$

where each component is defined as follows.

State Space S The state at time step t, denoted by $s_t \in S$, represents the current traffic condition at the intersection. In this work, the state is encoded as a four-dimensional vector:

$$s_t = [q_N(t), q_S(t), q_E(t), q_W(t)] \tag{4}$$

where $q_N(t)$, $q_S(t)$, $q_E(t)$, and $q_W(t)$ denote the number of vehicles waiting in the North, South, East, and West incoming lanes, respectively. This representation captures congestion levels in all directions and serves as the input to the Deep Q-Network.

Action Space A The action space consists of a discrete set of traffic signal phases:

$$A = \{a_1, a_2, \ldots, a_6\} \tag{5}$$

where each action corresponds to one of the following signal configurations:

- Green for North only
- Green for South only
- Green for East only
- Green for West only
- Green for North–South directions
- Green for East–West directions

At each decision point, the agent selects an action $a_t \in A$, which determines the active traffic signal phase during the next control interval.

Transition Function T The state transition function is defined as

$$T(s_{t+1} \mid s_t, a_t) \tag{6}$$

which specifies the probability of transitioning from state s_t to state s_{t+1} after taking action a_t. In this environment, transitions are governed by traffic dynamics such as vehicle arrivals, departures, and intersection movement rules. These dynamics are stochastic due to random vehicle generation and the presence of emergency vehicles.

The transition model is not explicitly known to the agent and is learned implicitly through interaction with the environment.

Reward Function R The reward function evaluates the effectiveness of a selected action in improving traffic flow. The immediate reward at time t is defined as:

$$r_t = \frac{2\,(Q_{t-1} - Q_t) + (W_{t-1} - W_t)}{\max(1, N_t)} \tag{7}$$

where:

- Q_t is the total queue length at time t,
- W_t is the cumulative waiting time of all vehicles,
- N_t is the number of vehicles currently present in the system.

This reward formulation encourages reductions in congestion and waiting time. Positive rewards indicate improved traffic conditions, while negative rewards reflect increased congestion.

Discount Factor γ The discount factor $\gamma \in (0, 1)$ determines the importance of future rewards. In this work, the value is set to:

$$\gamma = 0.959 \tag{8}$$

which balances immediate traffic improvements with long-term congestion reduction.

Objective Function The objective of the agent is to learn an optimal policy π^* that maximizes the expected discounted cumulative reward:

$$\pi^* = \arg\max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{9}$$

This optimal policy defines the best traffic signal control strategy over long-term operation.

Deep Q-Learning Approximation Due to the continuous nature of the state space, the action-value function is approximated using a neural network parameterized by θ:

$$Q(s, a; \theta) \approx Q^*(s, a) \tag{10}$$

The network parameters are updated by minimizing the temporal-difference loss:

$$\mathcal{L}(\theta) = E\left[\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta)\right)^2\right] \tag{11}$$

where θ^- denotes the parameters of the target network.

Implementation

This section describes the practical realization of the proposed smart traffic signal control system using Deep Reinforcement Learning. The focus of the implementation is on integrating a traffic simulation environment with a Deep Q-Network (DQN) agent, enabling real-time decision-making and online learning.

Tools and Libraries

The complete system was implemented using **Python 3**, which was selected due to its simplicity, wide adoption in artificial intelligence research, and strong support for numerical computing and deep learning frameworks.

The traffic environment and visualization were developed using the **PyGame** library. PyGame enables real-time rendering of roads, vehicles, traffic signals, and signal phase transitions.

NumPy was used extensively for numerical operations, including state representation, queue length computation, and reward calculation.

The Deep Q-Network was implemented using **PyTorch**, a popular deep learning framework. PyTorch was chosen for its flexibility in defining neural network architectures .

Hyperparameters

The learning performance of the DQN agent depends heavily on the choice of hyperparameter. The values used in this implementation were selected based on commonly accepted practices in the reinforcement learning literature and through empirical experimentation.

Table 1: Hyperparameters used in the DQN-based traffic control system

Parameter	Value
Learning Rate	0.001
Discount Factor (γ)	0.95
Batch Size	64
Replay Buffer Size	6000
Initial Exploration Rate (ε)	1.0
Minimum Exploration Rate ($\varepsilon_{\min}$)	0.05

The learning rate controls the magnitude of weight updates during training and was set to a small value to ensure stable convergence. The discount factor γ determines the importance of future rewards and was chosen to encourage long-term traffic optimization rather than short-term gains.

The batch size defines the number of experiences sampled from the replay buffer during each training step, balancing computational efficiency and learning stability. The replay buffer size limits the number of stored experiences while maintaining sufficient diversity for effective training.

An epsilon-greedy exploration strategy was employed, where the exploration rate ε starts at 1.0 to promote exploration during early training and gradually decays to a minimum value of 0.05, allowing the agent to increasingly exploit learned traffic signal policies.

Results

This section presents the experimental outcomes obtained from the implementation of the Deep Q-Network (DQN) based smart traffic signal control system. The performance of the system is evaluated using quantitative reinforcement learning metrics as well as qualitative observations from the simulation environment.

Training Setup

The traffic control agent was trained in a custom PyGame-based simulation environment representing a four-way road intersection. Each training episode corresponds to one simulated day, consisting of a fixed number of time steps. At every decision interval, the agent selects a traffic signal phase based on the current traffic state, receives a reward, and updates its policy accordingly.

The system was trained for multiple consecutive days in a single continuous run. To ensure consistent and interpretable results, all random number generators (Python, NumPy, and PyTorch) were seeded, allowing the same traffic arrival patterns and learning dynamics to be reproduced across executions.

Evaluation Metrics

The performance of the proposed system was evaluated using the following metrics:

- **Average Daily Reward**: This metric represents the mean reward accumulated by the agent over all decision steps in a simulated day. It reflects how effectively the agent reduces traffic congestion and waiting times.
- **Average Vehicle Waiting Time**: The average number of time steps vehicles spend waiting at red signals before crossing the intersection. Lower values indicate smoother traffic flow.
- **Exploration Rate (ε)**: The exploration parameter of the epsilon-greedy policy, which decreases over time and reflects the agent's transition from exploration to exploitation.

Quantitative Results

Table 2 summarizes the learning performance of the agent over several training days.

Table 2: Daily performance statistics of the DQN-based traffic control system

Day	Avg Daily Reward	Avg Waiting Time	ε
1	-43.44	345884.99	1.00
2	-0.78	264194.95	0.90
3	3.31	131757.45	0.81
4	0.85	101145.89	0.73
5	0.55	63525.92	0.66
6	-0.42	77608.13	0.59
7	0.14	62753.91	0.53

The results show a clear improvement in system performance as training progresses. On the first day, the agent exhibits a strongly negative average reward due to purely exploratory behavior, resulting in inefficient signal switching and severe congestion. As training continues, the average daily reward increases and becomes positive, indicating that the agent has learned effective traffic signal policies.

The reduction in average waiting time across days further confirms the agent's ability to improve traffic flow. Although occasional negative rewards are observed (e.g., Day 6), these fluctuations are expected in reinforcement learning systems and are primarily caused by residual exploration and variations in traffic demand.

Qualitative Observations

In addition to numerical metrics, visual inspection of the simulation provided valuable insights. During early training stages, vehicles frequently accumulated at red lights, leading to long queues and stop-and-go traffic. As training progressed, the agent began prioritizing directions with higher traffic density, resulting in smoother vehicle movement and reduced idle time.

Emergency vehicles were also observed to receive preferential treatment due to the reward structure, allowing them to pass through intersections with minimal delay. This behavior emerged naturally without explicit rule-based control, highlighting the adaptability of the reinforcement learning approach.

Discussion

The experimental results confirm that reinforcement learning can effectively control traffic signals in a dynamic and uncertain environment. During the initial training phase, the agent exhibits suboptimal performance due to random exploration driven by a high exploration rate. As training progresses, the agent gradually learns to prioritize heavily congested lanes and to coordinate opposing traffic directions more efficiently.

Minor fluctuations in the reward values were observed throughout training. These variations can be attributed to several factors, including random vehicle arrivals, emergency vehicle priority overrides, and the continuous (non-episodic) nature of the traffic environment. Despite these challenges, the agent demonstrates stable learning behavior and progressively improves overall traffic performance by reducing congestion and average waiting times.

Conclusion

This project successfully demonstrates the application of **Deep Q-Learning (DQN)** to the problem of smart traffic signal control. The proposed system dynamically adapts traffic signal phases based on real-time traffic conditions, enabling more efficient traffic flow compared to fixed-time control strategies.

The key objectives of adaptive signal control, real-time visualization, and learning performance analysis were achieved effectively. The results highlight the potential of reinforcement learning techniques for intelligent transportation systems and smart city applications, particularly in environments characterized by uncertainty and dynamic demand.

References

1. Pieter Abbeel, Dan Klein, and Sergey Levine. *CS188: Introduction to Artificial Intelligence – Reinforcement Learning II*. University of California, Berkeley. Lecture slides, 2025.
2. PyGame Documentation. Available online.
3. PyTorch Official Documentation. Available online.

Appendices

Appendix A: Source Code

The complete source code for the proposed traffic signal control system is submitted separately.