Autonomous Cybernetic Multi-Agent System for Traffic Intersection Control

Universidad Distrital Francisco José de Caldas

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Arlo Ocampo Gallego
Juan Ramos Ome

Supervisor: Eng. Carlos Andrés Sierra, M.Sc.

July 11, 2025

Contents

1. Abstract

This report presents the design and implementation of an autonomous cybernetic multi-agent system for traffic intersection control. The project aims to improve traffic flow and pedestrian safety using reinforcement learning (Q-learning and Deep Q-Networks) in a simulated environment of two interconnected intersections. The agents observe real-time data through virtual sensors, make decisions via a feedback control loop, and adapt behavior through learning mechanisms. Key results show a reduction in average vehicle queue length and pedestrian wait time, indicating the system's ability to optimize conflicting objectives. The integration of cybernetic principles such as feedback loops, homeostasis, and distributed control reinforces the robustness and scalability of the solution.

2. Introduction

- Definition of the problem.

  Develop an autonomous agent capable of learning and adapting to a simulated environment using reinforcement learning. The agent will monitor two intersections with traffic lights with multiple traffic participants (cars, motorcycles, and pedestrians).

  An autonomous agent will be designed to control and manage traffic lights in two intersections, with the goal of optimizing traffic flow with reinforcement learning in the simulation; the agent learns to respond to traffic lights by observing traffic conditions, in order to minimize waiting time and avoid traffic at the two points.

  2. Also delves into the analysis and refinement of the autonomous agent, utilizing concepts from dynamical systems. The primary goal is to enhance the agent's ability to adapt to complex traffic scenarios by employing tools such as Markov Decision Processes (MDP) and reinforcement learning, specifically Q-learning. Furthermore, the workshop explores the use of phase portraits to visualize system behavior and emphasizes the importance of stability and convergence in the agent's learning process. The design of advanced feedback mechanisms is also addressed to enable the agent to respond effectively to uncertain and dynamic environments.

- Objectives:

  Functional.

  1. Control and manage two intersections with traffic lights with multiply agent in the environment (pedestrians, vehicles).
  2. Optimize traffic flow with the implementation of reinforcement learning.
  3. Respond to different traffic flows and environment conditions (including initial conditions).

  Non-Functional

  1. Scale from 2 to up to 10 intersections.

3. Literature Review

3.1. Traditional traffic light control

Traditional traffic light control methods have been widely used for decades and are still prevalent in many cities around the world. These methods are typically based on fixed-time schedules or traffic-responsive strategies that adjust signal timings based on traffic volume or occupancy. Fixed-time schedules use predetermined signal timings that are set according to historical traffic patterns or the peak traffic volume of a specific area. For example, the fixed-time schedules are set using historical traffic demand to determine the time for each phase [1, 2]. The traffic-responsive strategies use updated time information that is set according to real-world traffic data. For example, Porche and Lafortune [3] and Cools et al. [4] implement self-organizing traffic lights using real-time traffic data. These methods can deal with highly random traffic conditions. One of the main drawbacks of fixed-time schedules is that they cannot adapt to changing traffic conditions, such as fluctuations in traffic volume or unexpected events. This often results in inefficient traffic flow, with long queues and delays and wasted time and fuel consumption. The problem with traffic-responsive strategies is that they are dependent on man-made rules for current traffic patterns, but do not consider the subsequent traffic conditions. In this way, it is unable to optimally adapt to the stochastic traffic flow dynamics.

3.2. Reinforcement learning-based traffic light control

Since traditional traffic light methods are not able to solve the multi-direction dynamic traffic light control problems comprehensively, there are more and more attempts using RL to deal with the problem. Traditional RL algorithms designed the state as discrete values of traffic conditions, such as the location of vehicles and the number of cars on the road [7]. To solve the problem to consider a larger state space, the algorithm of Deep Q learning is applied to take continuous variables into account. Deep Q learning sets up a deep neural network (DNN) to learn the Q function of RL from the traffic state inputs and the corresponding traffic system performance output. In this way, the state and action are associated with reward. The state design considers queue length and average delay, etc. The reward design also takes those variables into account. Nevertheless, these methods assume relatively static traffic environments, and hence far from real-world traffic conditions. Recently, Wei et al.Proposed an RL-based traffic light control model. The algorithm is tested with real-world traffic settings. In this paper, we follow the framework of Wei et al, but use the real-world traffic flow data from a road intersection in Hangzhou, China, which none of the existing studies has considered

4. Background

Urban traffic management has increasingly adopted intelligent systems capable of autonomous decision-making. Traditional fixed-time traffic lights often fail to adapt to dynamic and unpredictable traffic patterns, leading to congestion and inefficiency. Cybernetics offers a framework for systems that self-regulate through feedback, while reinforcement learning provides a computational tool for learning optimal actions over time. This project combines these two paradigms to develop a self-adaptive multi-agent system, where each agent controls one intersection based on local observations. The agents are trained using Markov Decision Processes (MDP), Q-learning, and DQN, leveraging tools such as SUMO, Gymnasium, and Stable-Baselines3 to simulate and refine their policies.

5. Objectives

The main objective of this project is to design, implement, and evaluate an autonomous cybernetic multi-agent system capable of controlling traffic lights at two intersections using reinforcement learning. The system aims to optimize vehicle and pedestrian flow by learning adaptive policies through interaction with a simulated environment.
Specific objectives include:

- Design and deploy autonomous agents: Use Q-learning and Deep Q-Networks (DQN) to manage two interconnected intersections under dynamic traffic conditions.
- Develop a simulation environment: Using SUMO, Gymnasium and Stable-Baselines3, modeling realistic scenarios that include pedestrian crossings, variable vehicle arrival rates, and environmental disturbances.
- Implement cybernetic feedback loops: Allow each agent to self-regulate and adapt to changing states through continuous observation, reward evaluation, and action selection.
- Define and optimize reward functions: Balance vehicle throughput, pedestrian service, and fairness in traffic signal allocation.
- Measure performance: Using key indicators such as the average length of queue in vehicles, the waiting time of pedestrians, and the total accumulated reward.
- Evaluate system robustness: In various scenarios, including rush hour loads, emergency events, and pedestrian surges.

Research Questions addressed:

- Can autonomous agents efficiently learn to control traffic lights in a multi-agent setting using reinforcement learning?
- How do cybernetic feedback principles enhance the adaptability and stability of such agents?
- What are the trade-offs between vehicle flow optimization and pedestrian service in dynamic traffic environments?
- How do Q-learning and DQN compare in performance for real-time traffic control tasks?

6. Scope

This section defines the boundaries and extent of the research carried out in the development of the autonomous cybernetic traffic control system. Clearly establishing what is included and excluded helps to focus the analysis and manage expectations regarding system capabilities, complexity, and outcomes. It also serves to frame the technical decisions made during design and implementation.

6.1. Inclusions:

- Simulation and control of two adjacent traffic intersections using reinforcement learning.
- Use of independent agents (Agent A and Agent B) trained with Q-learning and Deep Q-Networks.
- Implementation of feedback loops inspired by cybernetic control theory.
- Modeling of pedestrians, vehicles, sensor data, and actuator outputs.
- Evaluation of system performance in terms of traffic flow, pedestrian wait time, and reward accumulation.

6.2. Exclusions:

- Deployment in physical traffic infrastructure or integration with real-world traffic systems.
- Modeling of left-turn or U-turn maneuvers.
- Use of real-time camera feeds or physical sensors—only virtual simulation data is considered.
- Implementation of centralized coordination between agents (no communication protocols or shared memory).
- Legal, social, or ethical implications of AI in traffic control are not addressed.

By narrowing the scope to a simulated, two-intersection environment, the project ensures manageable complexity while still allowing meaningful experimentation and analysis of intelligent traffic control strategies.

7. Assumptions

The following assumptions were made to simplify modeling and simulation:

- Pedestrians and vehicles arrive following predefined stochastic distributions.

- The environment is fully observable through virtual sensors (no sensor noise or failure).
- Reinforcement learning agents operate with discrete state and action spaces.
- Intersections are isolated units controlled locally by agents without centralized coordination.
- Pedestrian behavior is limited to crossing requests and does not include jaywalking or group dynamics.
- All episodes are simulated under ideal network and computational conditions (no communication delays or hardware constraints).
- Emergency vehicle behavior and weather disturbances are introduced only in specific test scenarios.

8. Methodology

8.1. System Analysis

8.1.1 Actors

- Vehicles: Cars/motorcycles moving through intersections, provides traffic flow input, are controlled indirectly by traffic lights.
- Pedestrians: People who want to cross the street. May arrive at random times or during rush periods. Trigger pedestrian-specific light sequences and influence agent decisions.
- Cybernetic agents (controllers): Devices that control vehicle and pedestrian flow. Each intersection has its own. Core decision-makers; learn to manage traffic efficiently.

Table 1: Sensors

sensors	**What does the sensor do?**	**Data provided by the sensor**
Camera in the two traffic lights	counts stopped and moving vehicles	number of vehicles before, between and after traffic lights
status timer	time spent in a state (red, yellow, green)	time in seconds
vehicles camera	counts the vehicles that pass at change the state of the traffic light	average number of vehicles per unit of time
people detector	detects if there are people waiting	yes or no

Table 2: Actuators

Actuators	**what the actuator does**
first traffic light controller (A)	changes the state of the traffic light according to the agent (red, yellow, green)
second traffic light controller (B)	changes the state of the traffic light according to the agent (red, yellow, green))
timer	adjusts the time based on the defined parameters

8.1.2 Use Cases

Use case 1
Title: Optimizing vehicle flow between the two traffic lights.
Priority: High.
Estimate: 5 Days.
User story: As an intelligent traffic control agent, I want to learn to coordinate two consecutive traffic lights so that the flow of vehicles on the main road is optimized and the waiting time is minimized.
Acceptance Criteria:

- Given the agent is in a simulated environment during training.
- When it makes decisions to switch traffic lights.
- Then it learns to reduce traffic congestion and improve average travel time.

Use case 2
Title: Reducing the average wait time.
Priority: Medium.
Estimate: 3-4 Days.
User story: As an autonomous control system, I want to adjust the traffic light in real time based on traffic variation, so that the system remains efficient during both peak and low traffic hours.
Acceptance Criteria:

- Given traffic conditions vary over time.
- When the agent receives updated sensor observations.
- Then it modifies to maintain optimal the traffic flow without manual intervention.

Use case 3
Title: Ensuring pedestrian crossing safety
Priority: High
Estimate: 3 Days

User story:
As a pedestrian-aware traffic agent, I want to recognize when a pedestrian requests to cross, so that I can allow safe crossing while minimizing traffic disturbance.
Acceptance Criteria:

- Given a pedestrian presses the crossing button
- When the agent detects and schedules a crossing phase
- Then the pedestrian safely crosses with minimal vehicle disruption

Use case 4
Title: Learning from variable initial conditions
Priority: Medium
Estimate: 4 Days
User story:
As an adaptive traffic agent, I want to learn under different traffic start conditions, so that I generalize well and perform reliably in real-world dynamic scenarios.
Acceptance Criteria:

- Given the environment starts with different vehicle/pedestrian configurations
- When the agent begins learning
- Then it should converge to an effective policy across conditions

8.1.3 System Requirements

1. Inputs:
   (a) Traffic flow data: Number, speed and position of vehicles approaching each intersection.
   (b) Pedestrian request: Presence or request of pedestrian waiting to cross.
   (c) Traffic light status: Current state of each light at both intersections.
   (d) Environment conditions: (Optional) Time of day, weather, noise, emergencies.
   (e) Reward signals: Feedback (from reward functions) about the consequences of each action.

2. Outputs:
   (a) Traffic lights commands: Actions to change lights for each direction.
   (b) Agent state updates: Internal weights or policy values adjusted via reinforcements learning.
   (c) (optional) Log/telemetry data: Statistics for monitoring (waiting time, flow rate, reward score)

   Light states, timing signals

3. Constraints:

   (a) Time

   - Real time responsiveness: The system must act within short time intervals.
   - Learning horizon: Reinforcement learning should converge in a reasonable simulation time.

   (b) Space

   - Limited intersection area: Vehicles and pedestrians interact in a small 2D simulation area.
   - Agent memory: Each agent has limited memory/state size.

   (c) Safety

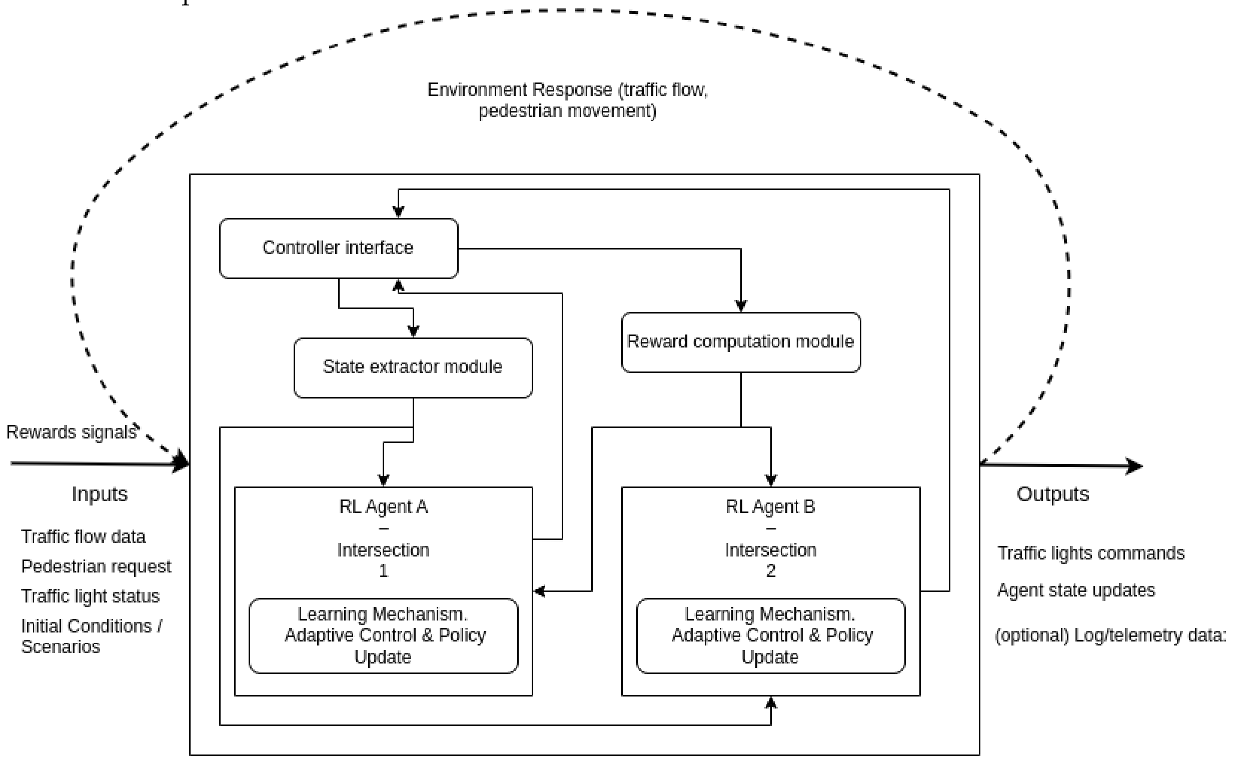
   - No simultaneous conflicts: Lights must avoid situations where both pedestrian and vehicles have green in the same path.
   - Minimum green/red duration: Enforced delay to prevent unsafe rapid switching.

Time, space, safety

8.2. System Design

8.2.1 High-Level Architecture

Describe components.


Environment Response (traffic flow, pedestrian movement)
Controller interface
State extractor module
Reward computation module
Rewards signals
Inputs
Traffic flow data
Pedestrian request
Traffic light status
Initial Conditions / Scenarios
RL Agent A – Intersection 1
Learning Mechanism. Adaptive Control & Policy Update
RL Agent B – Intersection 2
Learning Mechanism. Adaptive Control & Policy Update
Outputs
Traffic lights commands
Agent state updates
(optional) Log/telemetry data:

8.2.2 Feedback Loops

The core feedback mechanism of the system follows the cybernetic principle of continuous self-regulation through environmental interaction. Each agent (Agent A or Agent B) is embedded in a closed-loop where observations, decisions, and outcomes cycle continuously over time. This feedback loop integrates discrete modules explicitly modeled in the system architecture and visualized in the system-level black box diagram.

The process unfolds as follows:

1. **Inputs from Environment:** Traffic flow data, pedestrian requests, current traffic light status, and other contextual conditions are received from the SUMO simulation and passed into the *Controller Interface.*

2. **State Extraction:** The *State Extractor Module* transforms raw simulation data into structured observations that can be interpreted by the agents (e.g., vehicle counts per direction, binary pedestrian waiting flags).

3. **Decision Loop (Agent):** Each RL agent receives:

   - The current state (from the State Extractor).
   - The most recent reward signal (from the Reward Computation Module).

   Using these, the agent updates its internal state (policy or Q-values), selects an action (e.g., changing the light phase), and outputs a traffic light control command.

4. **Actuation via Controller Interface:** The selected action is transmitted to SUMO through the *Controller Interface*, which modifies the signal phase at the corresponding intersection.

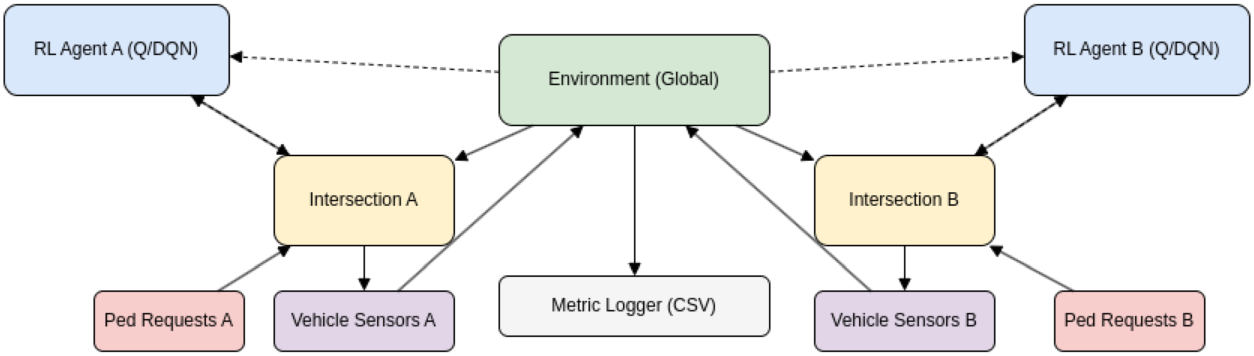
5. **Environment Transition and Feedback:** SUMO simulates the resulting vehicle and pedestrian dynamics. This generates:

   - A new environment state (to be extracted again).
   - A measurable outcome (e.g., vehicles crossed, pedestrians served, time wasted).

   These outcomes are interpreted by the *Reward Computation Module*, which computes scalar feedback signals to reinforce or penalize the agent's decision.

6. **Learning Cycle:** The feedback (reward) is routed back to the agent, completing the cybernetic loop. The agent updates its behavior through learning (e.g., Q-table update or gradient-based adjustment in a neural policy).

This feedback loop ensures that agents continuously adapt to changing traffic patterns and dynamic pedestrian behavior. Each component in the loop has a clearly defined role, and the feedback is not symbolic—it directly modifies the agent's future actions. This architecture embodies cybernetic self-regulation, not just by reacting to the environment but by adapting internal structure based on consequences.



8.2.3 Distributed Control

In this system, control is distributed across multiple agents, each associated with a specific intersection (Agent A and Agent B). These agents operate independently, without a central controller, but act in a shared environment simulated by SUMO. Each agent receives only local observations (vehicles and pedestrians at its own intersection) and decides on its own light phases.

While fully decentralized, implicit coordination emerges through environmental feedback. For example, if Agent A allows a large westbound vehicle flow, Agent B will observe the incoming traffic and adapt accordingly. This embodies distributed control in the cybernetic sense: self-organized regulation without global oversight.

Distributed control provides:

- **Robustness**: If one agent underperforms or fails, others still operate.
- **Scalability**: More intersections can be added without altering the control architecture.
- **Realism**: Reflects real-world smart infrastructure where control is localized.

8.2.4 Behavior Equations (Revised Model)

Each agent's internal behavior is modeled using time-dependent differential equations that evolve based on feedback from a stochastic environment. These equations do not dictate specific actions but represent internal behavioral tendencies that regulate how the agent prioritizes objectives.

- $x(t)$: Tendency to prioritize vehicle throughput
- $y(t)$: Tendency to prioritize pedestrian crossing
- $z(t)$: Tendency to avoid wasting green light time

Their evolution is governed by the following system of equations:

$$\frac{dx}{dt} = -\alpha_x(x(t) - x_0) + r_1(t) \tag{1}$$
$$\frac{dy}{dt} = -\alpha_y(y(t) - y_0) + r_2(t) \tag{2}$$
$$\frac{dz}{dt} = -\alpha_z(z(t) - z_0) + r_3(t) \tag{3}$$

Where:

- $\alpha_x, \alpha_y, \alpha_z$ are decay rates (self-regulation coefficients)
- x_0, y_0, z_0 are homeostatic baselines
- $r_1(t)$ increases with vehicles successfully crossing
- $r_2(t)$ increases with pedestrian waiting time (negative reinforcement)
- $r_3(t)$ increases when green lights are underutilized

These equations form a cybernetic feedback model in which each behavioral tendency is pulled toward a baseline unless modified by feedback from the environment, which is inherently stochastic.

8.3. Learning Model

8.3.1 Q-Learning Design

- State definition
- Action space
- Reward structure

8.3.2 Reward Function Table

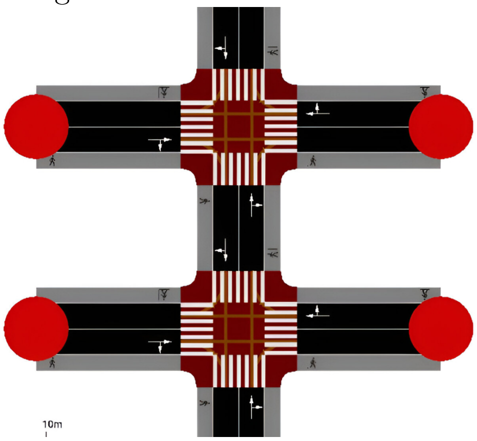
- Rewards for throughput
- Penalties for delays or collisions

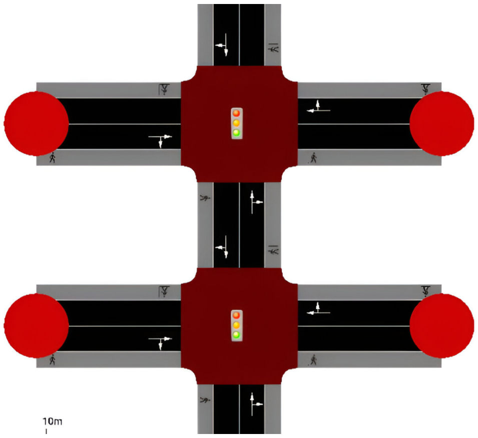
8.3.3 Transition to Deep Q-Network (DQN)

Discuss use of neural networks, experience replay, etc.

8.4. Simulation Environment

Next, the simulation environment is represented by two pictures that represent the two intersections and possible directions of each street, also in the left picture it is possible to identify pedestrian crossing walks and in the right picture visualize the location from the traffic light in each intersection.





8.5. Timeline

The following timeline presents an updated version of the previously submitted work plan. This revision incorporates refinements based on project progress, received feedback, and a clearer definition of development stages. While maintaining a weekly structure, the updated timeline introduces more detailed technical tasks and specific objectives for each phase, particularly in the implementation of the simulation environment, reinforcement learning agent training, and full system integration.

Week 1 – Introduction to Systems and Problem Analysis

- Introduction to System Theory, interactions, flows, synergy, emergent properties
- Overview of holistic approaches and system thinking
- Introduction to complex systems behavior and system engineering
- Scenario analysis: two intersections with traffic lights
- Identification of key actors (vehicles, sensors, pedestrians, environment)
- **Objective:** Understand and document the problem using a system thinking approach

Week 2 – Theoretical Foundations of AI and Control Systems

- Introduction to AI, intelligent systems, machine learning
- Explain main goals of AI, symbolic vs subsymbolic AI

- Introduction to feedback mechanisms, feedback loops, open vs closed loop systems
- Concept of homeostasis, dynamic systems, and chaos theory
- Review of supervised, unsupervised, and reinforcement learning
- Literature review of related AI applications in traffic management
- **Objective:** Connect system theory and AI principles to the project context

Week 3 – Environment and Gym Design

- Define environment states (vehicles, people, time)
- Define actions (traffic light changes)
- Logical structure of environment, rewards, sensors
- Begin implementation of the Gym environment
- **Objective:** Create the functional simulation space using Python and Gymnasium

Week 4 – Sensor Logic and Traffic Simulation

- Vehicle simulation programming
- Traffic light logic and timers
- Sensor modeling (zone counting, observation space)
- **Objective:** Establish simulation and observation infrastructure

Week 5 – Q-Learning Setup and Initial Training

- Implement basic RL agent with Q-table
- Training in multiple simple scenarios
- Visualize agent behavior and state transitions
- **Objective:** Initial training with Q-learning and basic performance review

Week 6 – Evaluate and Tune Q-Learning

- Analyze Q-learning performance
- Refine reward function
- Tune parameters and visualize learning improvements
- **Objective:** Improve the Q-learning agent through testing and feedback

Week 7 – Transition to Deep Q-Network (DQN)

- Introduce Stable-Baselines3
- Implement DQN agent
- Train and compare performance with Q-table agent
- **Objective:** Deploy the first DQN-based model

Week 8 – DQN Performance Analysis

- Analyze DQN results
- Adjust rewards, learning rates, and experience replay settings
- **Objective:** Stabilize and improve the DQN agent

Week 9 – Deep Dive into Agent Optimization

- Optimize hyperparameters
- Review impact of changes on traffic efficiency
- **Objective:** Refine agent decision-making capabilities

Week 10 – Multi-Agent and Edge Case Consideration

- Consider expansion to multi-agent settings
- Handle edge cases (pedestrian surge, sudden vehicle increase)
- **Objective:** Prepare system for more complex, realistic scenarios

Week 11 – Compare Learning Models

- Evaluate and compare Q-learning and DQN quantitatively
- Metrics: average wait time, traffic flow, agent decisions
- **Objective:** Provide evidence-based performance comparison

Week 12 – Incorporate Additional Complexity

- Add complexity: emergency vehicles, weather, time of day
- Modify agent behavior and reward structures
- **Objective:** Increase simulation realism and agent robustness

Week 13 – System Integration

- Integrate all modules: sensors, simulation, learning, control logic
- Begin system-wide testing
- **Objective:** Full system integration

Week 14 – Final Testing and Analysis

- Run simulations and record data
- Evaluate system performance and edge-case handling
- Final tweaks and preparations for documentation
- **Objective:** Validate system under all planned test conditions

Week 15 – Documentation and Delivery

- Final report (including system diagrams and code documentation)
- Presentation preparation
- Submit deliverables
- **Objective:** Deliver complete project with results and analysis

8.6. System Dynamics Analysis

8.6.1 Mathematical/Simulation Model

For the agent to learn to intelligently control traffic lights, it is necessary to represent mathematical the environment and its decisions. Two principal models that we are used for this: the Markov Decision Process (MDP) and the Q-learning algorithm.

The MDP allows the operation of the traffic system to be described as a series of diferent states (how many vehicles there are, the state of the traffic ligth, and actions like changing or maintaning the traffic light phase, and rewards like reducing the waiting time). This helps to define what the agent observes and what it can do.

Then the Q-learning is applied, a reinforcement learning algorithm that allows the agent to learn through trial and error. With the simulation the agent tests different actions, observes the results, and learns which decisions are best for improving vehicular flow. Over time, the agent learns a strategy that optimizes traffic light behavior.

MDP

Is a decision mathematical framework for describing decision-making problems in dynamic and uncertain environments, such as traffic.

$$MDP = (S, A, P, R) \tag{4}$$

Where:

- S: Describe the current situation of the environment, in these case, the traffic and traffic lights.
- A: They are the decisions that the agent can make in a given state.
- P: Describes the probability of the system moving from one state to another.
- R: It is a numerical value that represents how good an action was in a given state.

8.6.2 Discrete dynamic traffic model

Variables:

- x_t: number of vehicles waiting in a lane over time t.
- a_t: agent's action at time t (e.g., changing to green or maintaining the current state).
- λ_t: vehicle arrival rate over time t.
- μ_t: vehicle exit rate if the traffic light is green (flow).
- s_t: traffic light state at t (green = 1, red = 0).

Dynamic equation (state model):

$$x_{t+1} = x_t + \lambda_t - \mu_t \cdot s_t$$

This means:

- The number of vehicles at the next instant $t + 1$ It is the same as the ones that were there before,
- plus those who arrived(λ_t),
- less those who left, if the traffic light is green ($\mu_t \cdot s_t$).

Conditions and restrictions:

- $x_{t+1} \geq 0$: there can be no negative vehicles.
- $s_t \in \{0, 1\}$: traffic light can only be red (0) or green (1).

2.2 Phase Portraits or Diagrams

Phase portraits are a way to visualize the behavior of a dynamical system in its state space. The ”state space” is the set of all possible states that the system can be in. In our traffic control scenario, a ”state” could be defined by variables like:

- Number of vehicles before traffic light A
- Number of vehicles between traffic lights A and B
- Number of vehicles after traffic light B
- Average waiting time at each light
- Current phase of each traffic light (red, yellow, green)

And a probable phase portrait for this project could be a portrait where:

- X-axis = Number of vehicles before traffic light A
- Y-axis = Average waiting time at traffic light B

Attractors would represent desirable traffic flow patterns. For example:

- A stable attractor might be a point with ”moderate vehicle numbers before A” and ”low average waiting time at B.” This indicates efficient traffic flow.
- The agent’s goal is to guide the system towards these attractor states by making appropriate traffic light decisions.

In traffic control, chaos could manifest as:

- Unexpected surges in traffic that lead to oscillations in waiting times and vehicle numbers.

Swift changes could be present as:

- A long arrow might represent a sudden increase in waiting time due to a traffic incident.
- The agent’s actions (e.g., changing traffic light phases) should ideally produce smooth transitions in the phase portrait, avoiding abrupt changes that could disrupt traffic flow.

Phase portraits can show how the system responds to different inputs or conditions.

- Possible scenarios: ”high congestion”, ”low demand”, ”different arrival rates of vehicles,” etc.
- Comparing these phase portraits will reveal how the agent adapts (or fails to adapt) to varying inputs.

Stability is indicated by trajectories that remain within a bounded region of the phase portrait. Convergence is when trajectories approach an attractor over time.

- The agent should promote stability, preventing traffic conditions from spiraling out of control (e.g., unbounded queues).
- Convergence means that the agent learns to consistently guide the system towards optimal traffic flow.

8.7. Feedback Loop Refinement

8.7.1 Enhanced Control Mechanisms

Table 1: Additional sensors : Following the feedback from the previous workshop, these are the sensors that will help collect the best data on the road.

sensors	**What does the sensor do?**	**Data provided by the sensor**
Speed sensor	Measures the average speed of vehicles in each lane	Average speed (km/h or m/s)
Vehicle distance sensor	Estimate traffic congestion in real time	Average distance between vehicles
People waiting sensor	How long a person has been waiting	People waiting time
Number of cars passing per second	Measures the number of cars that pass the traffic light	Average number of cars passing per second

Table 2: More granular rewards : The reward functions that proposed in the previous workshop were highly complex. The new functions that will allow the agent to earn rewards and improve its efficiency will be shown below.

Reward signal	**Type of score reward**	**Advantage**
Fluency Reward	+1 . crossing vehicles	the number of vehicles crossing increases
Penalty for prolonged waiting of people	-1 . waiting people	Prevents the system from ignoring the people
You have to cross cars when it's green	-2 points if there is a green light but no vehicle is crossing	Improves efficient use of green time

These rewards will allow the agent to function well, allowing a constant flow of vehicles to avoid congestion and allowing the people to cross the street. In the first reward, we see that there is a reward for each vehicle that crosses the traffic light. However, a time limit will be set for waiting for the people (if there are any). If this limit is exceeded, points will begin to be deducted. If there are no people, the flow of vehicles will continue.

In the design of an intelligent traffic light control system, the system's outputs correspond to the actions taken by the agent and their direct effects on the environment. These include the change in traffic light state (green, yellow, or red), the duration of each phase, the resulting traffic flow (how many vehicles cross), the accumulated waiting time for pedestrians and vehicles, and the reward obtained for each action.

Several of these outputs not only impact the environment but are also fed back to the system as new inputs through sensors. For example, the number of vehicles crossing is recorded again by the crossing camera and speed sensor, the waiting time for pedestrians is measured by the waiting sensor, and the distance between vehicles is detected by the congestion sensor. In this way, the system establishes a closed information loop where each action of the agent modifies the environment, and these changes become new inputs for future decisions.

Additionally, there are internal feedback mechanisms within the agent itself. The agent maintains a history of rewards and transitions that directly influences the learning and updating of its decision policy. In approaches such as Q-learning or Deep Q-Networks (DQN), this feedback is implemented by updating action values or by continuously training a neural network based on past experiences. This allows the agent to progressively adjust its behavior to achieve better results.

Together, this interaction between outputs, inputs, and internal feedback ensures that the system evolves dynamically, learning from the consequences of its actions to optimize traffic flow and reduce waiting times for both vehicles and pedestrians.

8.7.2 Stability and Convergence

To evaluate the effectiveness of the traffic control agent, we define stability and convergence in operational terms: the agent should not only reduce congestion over time, but do so consistently across varying traffic conditions. We adopt a forecasting approach that translates these goals into concrete performance metrics and simulation-driven validation criteria.

Success Criteria

The agent is considered **successful** if it can:

1. Consistently maintain queue lengths below a system-defined threshold (e.g. no more than 10 vehicles per lane).

2. Reduce average waiting time by at least 25% relative to a fixed-time or random policy baseline.

3. Demonstrate robust behavior under disruptions such as traffic surges, pedestrian spikes, or emergency vehicle interventions.

4. Reach a learning plateau where policy changes yield marginal gains (convergence).

Quantitative Metrics

The following metrics will be collected over time to measure both stability and convergence:

- **Average Vehicle Wait Time** (seconds) [lower is better]
- **Vehicle Throughput** (vehicles/time window) [higher is better]
- **Queue Length** (vehicles per lane) [boundedness indicator]
- **Reward Moving Average** (per episode) [proxy for learning convergence]
- **Reward Variance** [indicator of policy stabilization]

Simulation-Based Evaluation

Convergence and stability will be empirically assessed using the SUMO-based simulation environment:

- **Episode-Based Monitoring:** We track reward, waiting time, and throughput across training episodes. Convergence is indicated by flattening reward curves and reduced variance.
- **Temporal Boundedness:** Queue lengths and delays are measured at fixed intervals (e.g. every 100 steps) to ensure they remain within a safety envelope.
- **Stress Testing:** We inject disturbances into the simulation (e.g. sudden traffic influx, sensor dropout, or emergency vehicle prioritization) and observe whether the agent returns to stable operation.

Scenario Forecasting

We define three primary test scenarios, each designed to elicit distinct challenges for the agent:

- **Rush Hour Load:** High vehicle inflow from all directions; success implies maintaining throughput without explosive queue growth.
- **Pedestrian Interruption:** Frequent crossings trigger red phases; success implies adapting phase durations to preserve flow.
- **Emergency Routing:** An emergency vehicle must be given priority; success implies fast, adaptive green-wave responses with minimal network disruption.

Each scenario is run for a fixed duration (e.g. 10,000 simulation steps), and the above metrics are logged to track the system's behavior over time. Statistical analysis (e.g. t-tests, moving averages) will be used to detect convergence and measure resilience under load.

8.8. Iterative Design Outline

The development of the autonomous traffic agent will follow an iterative cycle of design, evaluation, and refinement. Each iteration introduces new functionality or refines existing behavior based on experimental feedback. The process is grounded in control theory and reinforcement learning principles, with continuous performance monitoring and structured improvement.

8.8.1 Iteration Structure

Each iteration is composed of the following stages:

1. **Define Iteration Goal**: A specific system improvement or behavioral enhancement (e.g., reducing emergency vehicle delay, improving green wave coordination).
2. **Implement Update**: Modify the system to support the goal. This might include changes to data structures, network architecture, state representation, or reward functions.
3. **Simulate and Collect Data**: Run multiple episodes in SUMO and collect time series data, including vehicle throughput, queue length, and wait time.
4. **Evaluate Against Criteria** (see Evaluation Strategy): Use pre-defined metrics and success thresholds to assess whether the changes improve performance.
5. **Analyze Results**: Use statistical plots, error traces, or diagnostic heatmaps to identify emerging issues or confirm improvements.
6. **Decide Next Action**:
   - If metrics improve: consolidate changes and proceed to next goal.
   - If regressions occur: roll back or revise implementation.
   - If no change: test new hypotheses (e.g., modify exploration strategy).

8.8.2 Iteration Planning and Forecasting

To keep development goal-driven, we define forward-looking targets for upcoming iterations:

- **Iteration 1: Baseline Policy Integration**
  - Goal: Achieve stable learning curves using basic Q-learning or DQN
  - Output: Learning curve, average reward stabilization, bounded queue lengths
- **Iteration 2: Enhanced Temporal Awareness**
  - Goal: Improve short-term prediction using RNNs or LSTMs
  - Output: Lower variance in reward trajectory across episodes with cyclical demand

- **Iteration 3: Emergency Responsiveness**
  - Goal: Ensure emergency vehicles receive priority with no major throughput penalty
  - Output: Emergency response time < 30 steps; $\leq 10\%$ throughput drop
- **Iteration 4: Multi-Agent Coordination**
  - Goal: Design a basic message-passing protocol and test green-wave emergence
  - Output: Reduced queue length at corridor endpoints; increased throughput vs. independent agents

8.8.3 Mechanisms for Feedback and Change

- **Data Logging**: All episodes are logged using time series data structures that store:
  - Traffic state vectors
  - Agent actions
  - Rewards
  - Outcome metrics per episode
- **Adaptive Representations**: Later iterations may require shifting from flat state vectors to structured representations (e.g., graphs of intersections or LSTM-compatible sequences).
- **Advanced Algorithms (Planned)**:
  - Recurrent neural networks (RNNs) for temporal pattern modeling
  - Kalman filtering for state estimation and prediction under uncertainty
  - Thompson sampling or UCB for smarter exploration strategies
- **Toolchain Evolution**:
  - Deep learning frameworks: PyTorch 2.0.1
  - Simulation environment: SUMO with TraCI interface
  - Optional visualization: TensorBoard, custom matplotlib dashboards

8.8.4 Simulation Parameters and Scenario Design

Table 4: Simulation Parameters

Parameter	**Description**	**Example**
Simulation duration	Total time of training	30 min to 1 hour
Decision interval	Frequency of agent action	Every 30 seconds
Phase change latency	Minimum time before switching signals	30 seconds

Scenario Variations for Adaptability Testing:

- **High Congestion**: Ensures the agent avoids gridlock through phase balancing.
- **Low Demand**: Tests idle-phase avoidance and efficient cycle skipping.
- **Weather Conditions**: Simulated delays on vehicle movement; system must adapt phase durations to avoid unsafe switching.

8.9. Machine Learning Implementation

8.9.1 Algorithms and Frameworks

- **Q-Learning**:Q-learning is implemented as a tabular agent with discretized state representations including vehicle queues, pedestrian requests, pedestrian crossing timers, and current signal states. The agent uses the Bellman equation for Q-value updates and ϵ-greedy exploration.
- **Deep Q-Network (DQN)**:DQN uses a neural network to approximate the Q-function. It employs an experience replay buffer and target network for stable updates. Implemented in PyTorch, the model is trained incrementally each episode and generalizes better over larger state spaces.

8.9.2 Cybernetic Feedback Integration

The system uses a real-time feedback control loop where the environment's state—including queues, pedestrians, and signal status—is used by the agents to optimize future actions. The reward function incorporates:

- Vehicle throughput
- Pedestrian crossing success
- Penalization for blocking pedestrians
- Penalty for excessive right-turn signal duration

This feedback enables the agents to self-regulate under changing traffic demands.

8.10. Environment and System Design

8.10.1 Intersections and Flow Direction

There are two intersections, A and B, each with 4 directions: North, East, South, West. Vehicles may go straight or turn right. Intersections are connected:

- A[N] → B[S], A[E] → B[N] (right turn)
- B[S] → A[N], B[W] → A[S] (right turn)

8.10.2 Pedestrian Dynamics

Pedestrians arrive randomly from each direction. When a crossing is initiated, it blocks vehicle movement for a fixed number of steps (e.g., 3). Agents are not required to immediately serve pedestrians but are rewarded for timely service and penalized for mid-crossing interruptions.

8.10.3 Turn Rules

If a signal is kept green for right-turn traffic for too long (beyond 3 steps), the agent is penalized, promoting fairness and preventing congestion buildup in one direction.

9. Results and Analysis

9.1. Total Reward (Q-learning)

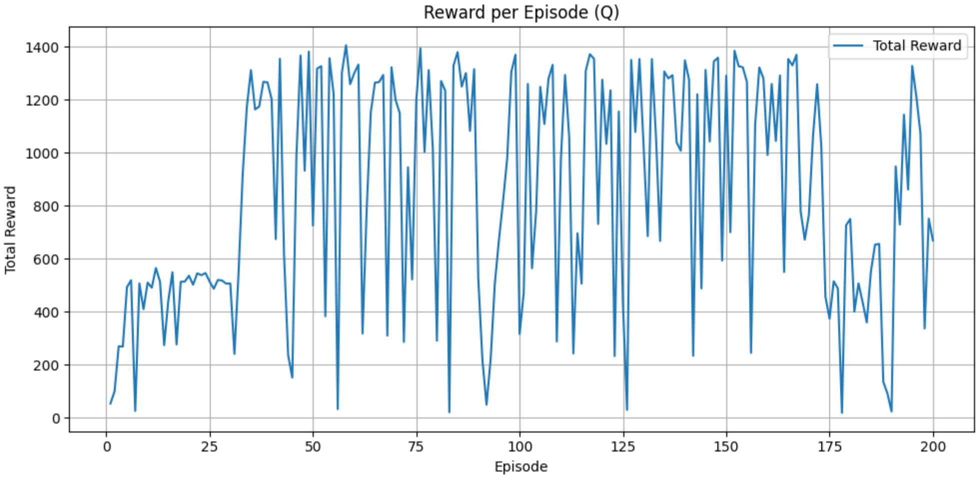


Figure 1: Total Reward per Episode – Q-learning

This graph shows the total reward obtained by the agent in each training episode. An initial increasing trend is observed, indicating that the agent is learning. Then, there is a high variability in the rewards, with high-performing episodes alternating with low-performing ones. This could be due to ongoing exploration or a highly dynamic environment.

Interpretation: The agent manages to learn effective policies but has not fully converged or is affected by environmental variability

9.2. Average Vehicle Queue Length

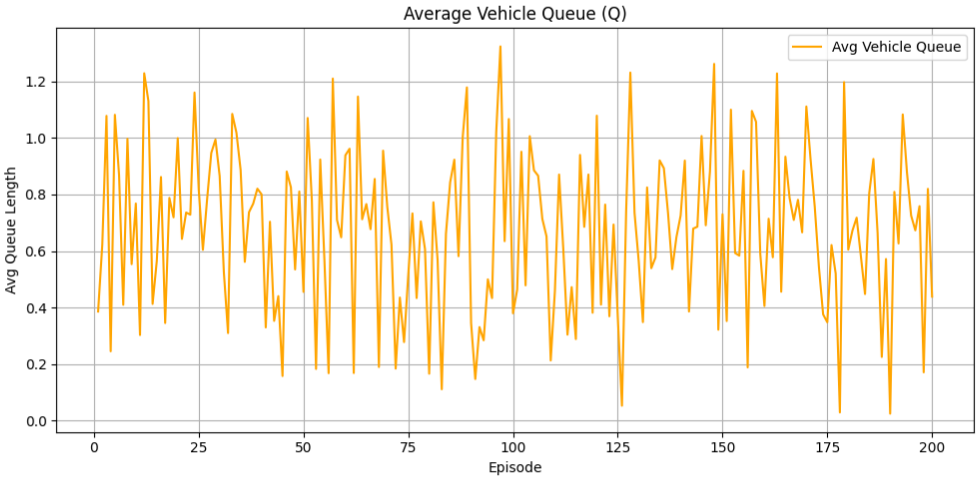


Figure 2: Average Vehicle Queue – Q-learning

This graph represents the average vehicle queue length per episode. The queue length remains between 0.2 and 1.2, indicating that on average, there is no severe congestion. The fluctuations may be caused by variations in traffic demand or suboptimal decisions in some episodes.

Interpretation: The system maintains a relatively short vehicle queue, although some episodes show higher accumulation.

9.3. Pedestrians Served

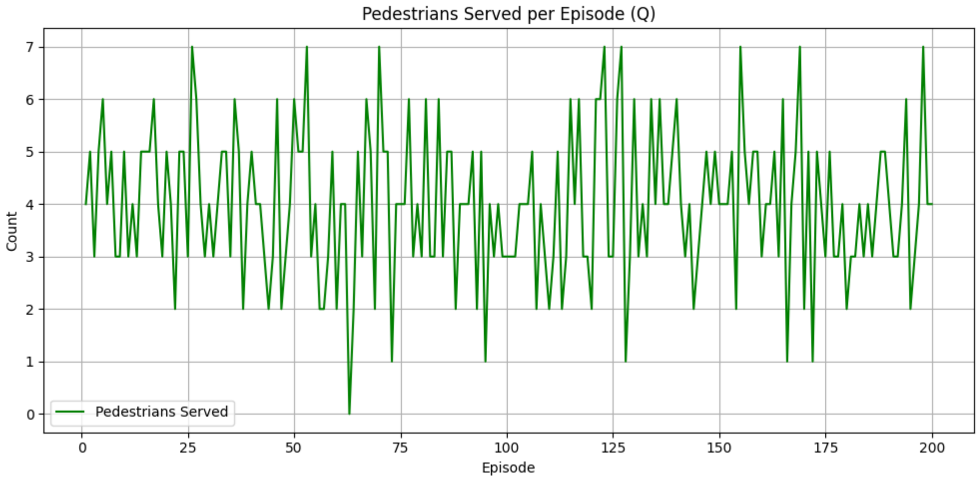


Figure 3: Pedestrians Served per Episode – Q-learning

This graph shows how many pedestrians were served (i.e., allowed to cross) in each episode. In general, the count stays between 3 and 6, indicating a good level of service. There are episodes where the number drops to 0 or 1, suggesting that pedestrians were sometimes not prioritized.

Interpretation: The agent attempts to balance vehicle traffic with pedestrian service, although it does not always succeed.

9.4. Average Pedestrian Wait Time

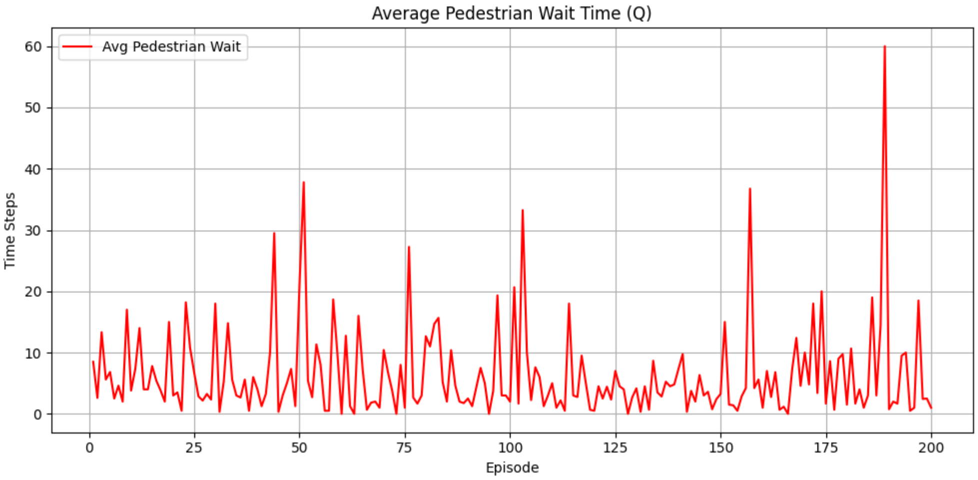


Figure 4: Average Pedestrian Wait Time – Q-learning

This graph presents the average pedestrian wait time per episode. Most wait times are below 10 time steps, which is positive. However, there are spikes exceeding 30 or even 60, indicating that pedestrians had to wait too long in certain episodes.

Interpretation: While average wait times are generally low, there are episodes with poor pedestrian signal management.

9.5. Interpretation

The agent learns to balance vehicle flow and pedestrian service. Early episodes show inconsistent behavior, but over time the system improves:

9.6. Test Metrics per Episode – Q-learning

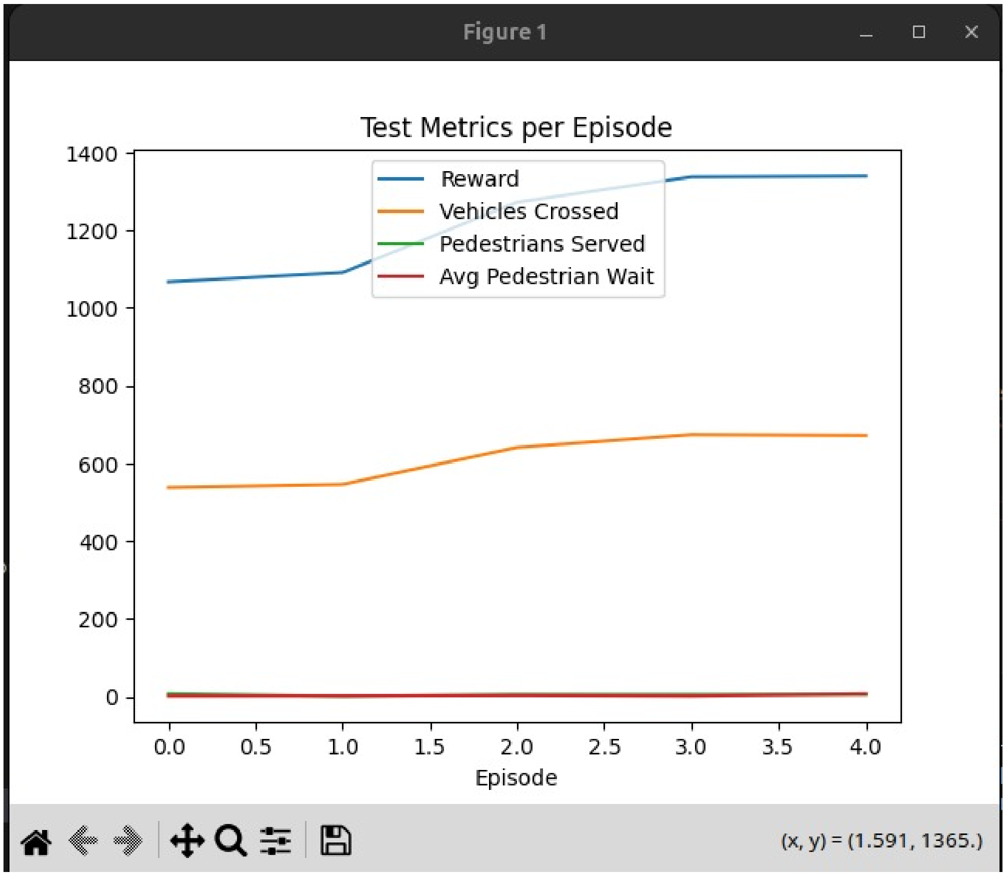


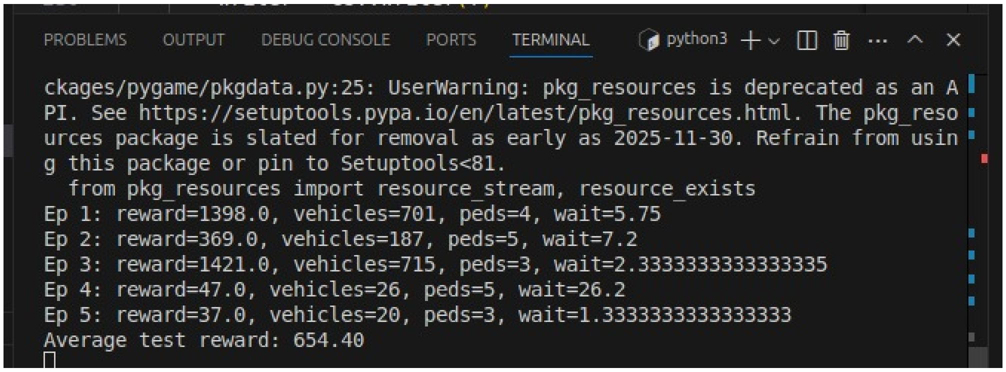
Figure 5: Pedestrians Served per Episode – Q-learning

Figure 5: Test Metrics per Episode – Q-learning
This graph displays various performance metrics of the Q-learning agent during the test phase, evaluated over 5 episodes.

- **Reward (blue line):** The total reward shows a consistent increase, indicating that the agent performs better over time by making more optimal decisions.
- **Vehicles Crossed (orange line):** The number of vehicles successfully crossing the intersection also increases across episodes, reaching a plateau. This reflects improved traffic flow management by the agent.
- **Pedestrians Served (green line):** The number of pedestrians served remains relatively stable with a slight upward trend, showing the agent maintains attention to pedestrian crossing needs.
- **Average Pedestrian Wait (red line):** The average waiting time for pedestrians remains low and consistent throughout the test, suggesting that the system prioritizes minimizing pedestrian delays.

Interpretation: The agent generalizes well during the test phase, maintaining high rewards, improving traffic throughput, and ensuring low pedestrian wait times. These results reflect a good balance between efficiency and fairness in the learned policy.

9.7. Terminal Output – Test Episode Statistics (Q-learning)



```
ckages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an A
PI. See https://setuptools.pypa.io/en/latest/pkg_resources.html. The pkg_reso
urces package is slated for removal as early as 2025-11-30. Refrain from usin
g this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
Ep 1: reward=1398.0, vehicles=701, peds=4, wait=5.75
Ep 2: reward=369.0, vehicles=187, peds=5, wait=7.2
Ep 3: reward=1421.0, vehicles=715, peds=3, wait=2.3333333333333335
Ep 4: reward=47.0, vehicles=26, peds=5, wait=26.2
Ep 5: reward=37.0, vehicles=20, peds=3, wait=1.3333333333333333
Average test reward: 654.40
```

Figure 6: Figure 6: Terminal Output – Test Episode Statistics (Q-learning)

- **Episode 1:** Achieved the highest reward (1398.0) with 701 vehicles crossed, 4 pedestrians served, and a low pedestrian wait time of 5.75 time steps.
- **Episode 2:** Performance declined with a reward of 369.0, 187 vehicles crossed, 5 pedestrians served, and a slightly higher pedestrian wait of 7.2.
- **Episode 3:** Performance recovered with a reward of 1421.0, 715 vehicles crossed, 3 pedestrians served, and a low wait time of 2.33.
- **Episode 4:** A notable drop in reward (47.0), with only 26 vehicles crossed and a high pedestrian wait time of 26.2.
- **Episode 5:** Lowest reward (37.0), with 20 vehicles, 3 pedestrians served, and the lowest pedestrian wait time (1.33).

The **average test reward** across all episodes is **654.40**, suggesting overall moderate agent performance with high variability between episodes.

Interpretation: The results show that while the agent is capable of achieving high-performance episodes, it lacks full consistency. Some episodes reach near-optimal performance, while others collapse, indicating potential instability or sensitivity to initial conditions in the environment.

9.8. Limitations

Despite its achievements, the project presents several limitations:

- Scalability: While designed for scalability, the current implementation handles only two intersections. Scaling to a city-wide network may require more sophisticated coordination mechanisms.
- Learning Stability: The Q-learning agent exhibits high reward variability in some episodes, suggesting incomplete convergence or sensitivity to environment dynamics.
- Simplified Dynamics: The simulation abstracts real-world noise, sensor failures, unexpected driver/pedestrian behavior, and road geometry.
- Pedestrian Logic: The current model does not simulate pedestrian crowd dynamics or adaptive behavior based on traffic conditions.
- Reward Design: The reward function requires manual tuning, and minor changes can significantly impact agent behavior.
- Computational Cost: Deep Q-Network training is computationally intensive and sensitive to hyperparameter configuration.

By acknowledging these constraints, future iterations can target improvements such as robust coordination, real-world deployment strategies, and advanced learning algorithms.

10. Conclusion

10.1. Conclusions

- The Q-learning agent demonstrates a clear learning progression, achieving increasingly higher total rewards across training episodes. Despite the variability during training, the test phase reveals a more stable and optimized behavior, characterized by higher vehicle throughput, consistent pedestrian service, and low average pedestrian wait times. These results indicate that the agent successfully learned to balance efficiency and fairness, adapting its policy to serve both vehicles and pedestrians effectively.
- The traffic control system based on Q-learning proves to be efficient and adaptive. During training, although fluctuations were observed—especially in pedestrian service and vehicle queue lengths—the system gradually stabilized. In the test phase, the agent maintained high performance with minimal pedestrian delays and stable metrics, confirming good generalization of the learned policy. The model thus presents a viable solution for real-time, multi-agent intersection control.

10.2. Future Work

- Add lane-specific vehicle types (e.g., priority/emergency lanes)
- Enable learning of pedestrian-light coordination strategies
- Extend environment to a 4-way grid of intersections

- Introduce stochastic demand patterns (rush hours, peak loads)
- Use shared or communicating agents for cooperative planning

References

[1] A. J. Miller, "Settings for fixed-cycle traffic signals," Journal of the Operational Research Society, vol. 14, no. 4, pp. 373–386, 1963.

[2] F. Dion, H. Rakha, and Y.-S. Kang, "Comparison of delay estimates at under-saturated and oversaturated pre-timed signalized intersections," Transportation Research Part B: Methodological, vol. 38, no. 2, pp. 99–122, 2004.

[3] I. Porche and S. Lafortune, "Adaptive look-ahead optimization of traffic signals," Journal of Intelligent Transportation System, vol. 4, no. 3-4, pp. 209–254, 1999.

[4] S.-B. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights: A realistic simulation," Advances in applied self-organizing systems, pp. 45–55, 2013.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[6] A. Perera and P. Kamalaruban, "Applications of reinforcement learning in energy systems," Renewable and Sustainable Energy Reviews, vol. 137, p. 110618, 2021.

[7] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown Toronto," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 3, pp. 1140–1150, 2013.