



Higher School of Computer Science - Sidi Bel Abbès

Specialty: Artificial Intelligence and Data Science

Project Report

Traffic Control Agent Using Deep Reinforcement Learning

Prepared by:

Hamdi Aya
Amieur Zineb Ichraq
Meflah Yousra

Supervisor:

Mr. Malki Abdelhamid

Academic Year: 2024 – 2025

Contents

1	Acknowledgment	3
2	Abstract	4
3	Introduction	5
4	Problem Statement & Motivation	6
5	Environment Description:	7
5.1	The Intersection Setup	7
5.2	Traffic Light Logic Description	9
5.3	Traffic Generation Mechanism	11
6	Reinforcement Learning Concepts	12
6.1	Deep Q-Network (DQN)	12
6.2	Double DQN	12
6.3	Dueling DQN	13
6.4	Target Network Updates	13
6.5	Experience Replay	13
6.6	Prioritized Experience Replay (PER)	13
7	Methodology	14
7.1	First Approach – Environment & MDP Design	14
7.2	Second Approach: Controlled Lanes as State with Dynamic Phase Durations	36
7.3	Third Approach: Throughput-Based Phase Control	42
8	Discussion	46
9	Challenges in Reinforcement Learning Framework	47
10	Conclusion	48

List of Figures

1	*	20
2	*	20
3	*	20
4	*	20
5	Training metrics for the DQN agent -version1-	20
6	Comparison of agent and default traffic control performance over episodes	21
7	*	24
8	*	24
9	*	24
10	*	24
11	Training metrics for the Double DQN + Dueling DQN agent - Version 2	24
12	Comparison of agent and default traffic control performance over episodes for Version 2	25
13	*	28
14	*	28
15	*	28
16	*	28
17	Training metrics for the DQN agent -version3-	28
18	Comparison of agent and default traffic control performance over episodes	29
19	*	32
20	*	32
21	*	32
22	*	32
23	Training metrics for the DQN agent -version4-	32
24	Comparison of agent and default traffic control performance over episodes	34
25	*	40
26	*	40
27	*	40
28	*	40
29	Training metrics for the DQN agent -version5-	40
30	Comparison of agent and default traffic control performance over episodes	41
31	*	44
32	*	44
33	*	44
34	*	44
35	Training metrics for the DQN agent -version6-	44
36	Comparison of agent and default traffic control performance over episodes	45

1 Acknowledgment

We would like to express our sincere gratitude to our mentor, Mr. Malki Abed El Hamid, for his continuous support, valuable feedback, and encouragement throughout the development of this project. His guidance and availability played a key role in helping us understand the challenges and possibilities of Reinforcement Learning in real-world applications.

2 Abstract

This project presents a reinforcement learning (RL)-based approach to intelligent traffic signal control using the Simulation of Urban MObility (SUMO) environment. The aim is to dynamically optimize traffic light phases at a junction to reduce congestion and improve vehicle flow. We develop and evaluate three progressively advanced environment designs tailored to train an RL agent.

The first approach uses a traditional fixed-duration control strategy, where the agent selects a green phase, and each phase lasts for a constant duration. The state representation is based on the queue lengths and waiting times from incoming edges. The reward function combines negative waiting time, queue length, and positive throughput, encouraging efficient traffic movement.

In the second approach, we introduce dynamic phase durations, allowing the agent to choose both the phase and a duration from a predefined set. The state is redefined using controlled lanes (i.e., lanes affected by the current green phase), offering a more granular view of local traffic conditions. This version enables more adaptive behavior compared to fixed-timed signals.

The third and most adaptive approach replaces time-based transitions with throughput-based control. Here, the agent selects a phase, and the environment transitions only when a defined number of vehicles have passed or a maximum duration is reached. The state remains lane-based but incorporates time-since-last-switch, enabling the agent to learn when to retain or change a phase. The reward function further emphasizes vehicle throughput while penalizing overall delay.

Experimental results demonstrate that each successive environment design allows the agent to develop more responsive and efficient traffic control policies. The throughput-based method shows significant improvements in reducing waiting times and increasing junction throughput, validating the benefit of integrating real-time traffic flow data into the decision-making process.

3 Introduction

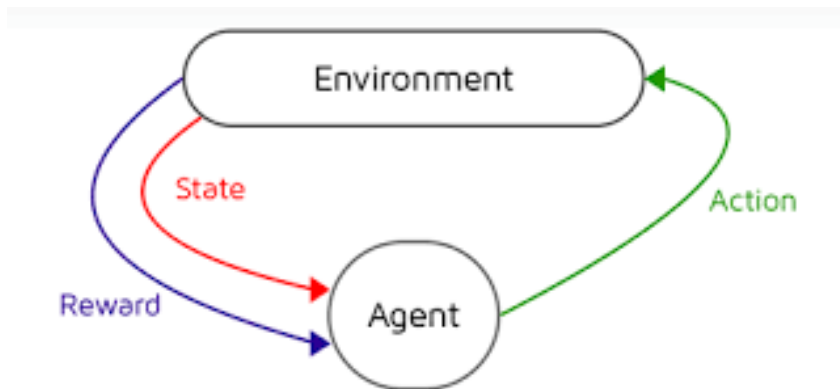
The rapid urbanization and increasing number of vehicles in modern cities have made efficient traffic management a critical challenge. Traditional traffic control systems, often based on static timing plans or manually tuned heuristics, struggle to adapt to dynamic traffic conditions, leading to congestion, increased travel times, and environmental impact. As a result, there is a growing demand for intelligent, adaptive solutions capable of responding in real time to changing traffic patterns.

Reinforcement Learning (RL), a subfield of machine learning, has emerged as a powerful framework for solving sequential decision-making problems. Unlike supervised learning, RL allows an agent to interact with its environment and learn optimal policies through trial and error, guided by a reward signal. Many control problems — including traffic signal optimization — can be modeled as Markov Decision Processes (MDPs).

An MDP provides a mathematical framework for modeling decision-making environments. It consists of a finite set of states (e.g., the current traffic situation at an intersection), a set of actions (e.g., change or maintain the traffic signal phase), a transition function (defining how the state evolves after each action), and a reward function (quantifying the benefit or cost of an action in a given state). The goal of the RL agent is to learn a policy that maximizes the cumulative reward over time. This structure makes MDPs well-suited for traffic control tasks, where decisions must be made continuously based on the current conditions.

In this project, we explore the use of Deep Q-Networks (DQN), a deep reinforcement learning algorithm, to train an agent capable of dynamically controlling traffic lights in a simulated urban environment. The simulation is conducted using SUMO (Simulation of Urban MObility), a widely-used, open-source traffic simulation platform. The agent is trained to minimize vehicle waiting times and improve overall traffic efficiency.

This report presents the problem formulation, the design of the simulation environment, the reinforcement learning methodology, and an evaluation of the system’s performance compared to traditional control strategies.

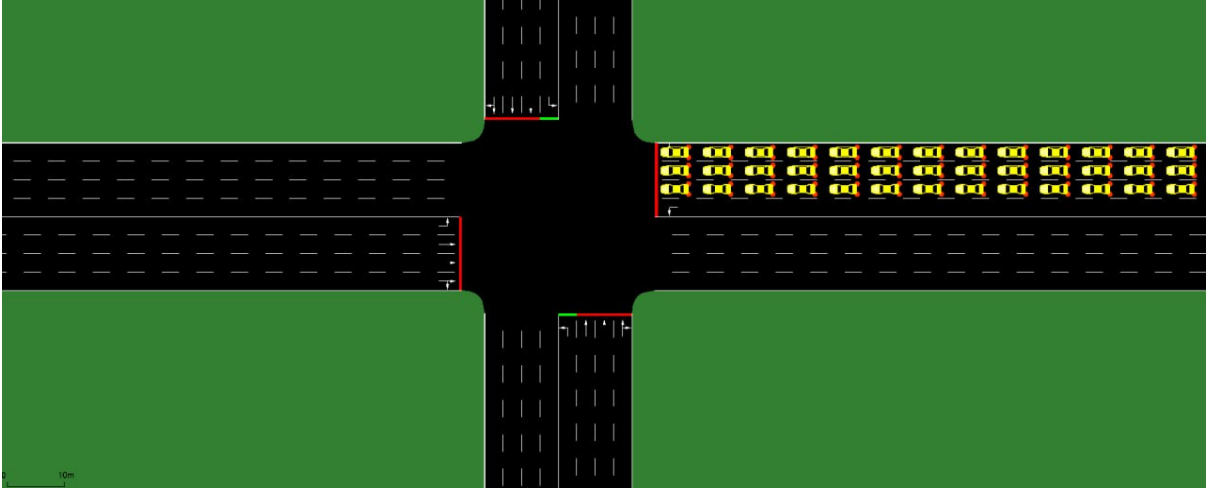


4 Problem Statement & Motivation

Urban traffic congestion remains a persistent challenge, especially at busy intersections where inefficient traffic light control can result in long vehicle queues, extended waiting times, and overall reduced traffic flow efficiency. In traditional traffic signal systems, light phases are often predefined and operate on fixed cycles, regardless of real-time traffic conditions. This static approach can lead to suboptimal scenarios — for example, maintaining a green light for 30 seconds even when the lane is already empty, or forcing vehicles to wait despite an opportunity to proceed.

In our project, we focus on a single intersection with four incoming roads, each comprising multiple lanes for straight, right-turn, and left-turn movements. The primary traffic challenges observed in such a setting include long waiting times at red lights and inefficient usage of green phases, especially under fluctuating traffic loads. These inefficiencies not only waste time but also contribute to fuel consumption and environmental pollution due to vehicle idling.

Conventional rule-based or fixed-timing traffic control methods lack adaptability and fail to respond to real-time traffic dynamics. As traffic patterns change throughout the day, a more intelligent system is needed to optimize traffic flow based on current conditions. This motivated us to explore Reinforcement Learning (RL), specifically using Deep Q-Networks (DQN), which allow an agent to learn optimal control policies through direct interaction with the environment. Unlike predefined rules, an RL-based controller can adaptively adjust signal phases to minimize waiting times and queues, ultimately improving the efficiency of urban traffic management.



5 Environment Description:

5.1 The Intersection Setup

The environment simulates a four-way urban intersection with two main crossing roads running in the **north-south** and **east-west** directions. Each approach to the intersection consists of four lanes numbered **0 through 3**.

The lane numbering and permitted maneuvers are as follows:

- **Lane 0:** Dedicated to right turns and also allows straight movement.
- **Lanes 1 and 2:** Dedicated to straight movement only.
- **Lane 3:** Dedicated to left turns only.

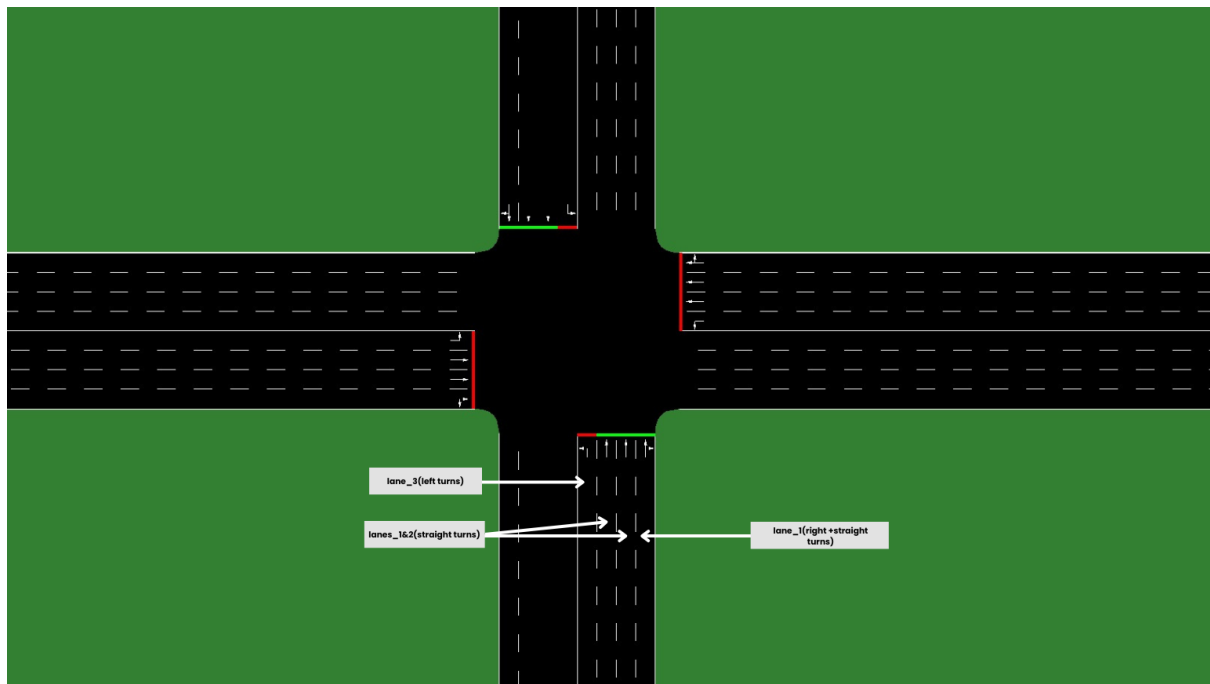
Roads and Edges

Each road has defined **incoming** and **outgoing edges** that represent vehicle flow to and from the intersection's traffic light node:

- **Incoming edges** represent lanes leading vehicles into the intersection's stop line (e.g., N2TL for northbound approach to the traffic light).
- **Outgoing edges** represent lanes leading vehicles away from the intersection after crossing (e.g., TL2S for vehicles leaving towards southbound).

The four roads and their corresponding edges are:

Road Direction	Incoming Edge	Outgoing Edges
North (N)	N2TL	TL2S, TL2N, TL2E, TL2W
South (S)	S2TL	TL2S, TL2N, TL2E, TL2W
East (E)	E2TL	TL2S, TL2N, TL2E, TL2W
West (W)	W2TL	TL2S, TL2N, TL2E, TL2W



5.2 Traffic Light Logic Description

The traffic light system is configured with **eight phases**, cycling between **four green phases** and **four yellow phases** to control traffic flow safely and efficiently. The phases are designed to minimize conflicts, especially for vulnerable left-turn movements.

Phase Grouping and Objectives

- **Green Phases:** Control the flow of vehicles moving straight, right, or turning left from specified directions.
- **Yellow Phases:** Provide transition periods warning drivers to prepare to stop before the next green phase.

Detailed Green Phases

Phase	Description	Active Lanes	Active Directions
0	North-South Straight + Right	Lanes 0,1,2	North and South
2	North-South Left Turns	Lane 3	North and South
4	East-West Straight + Right	Lanes 0,1,2	East and West
6	East-West Left Turns	Lane 3	East and West

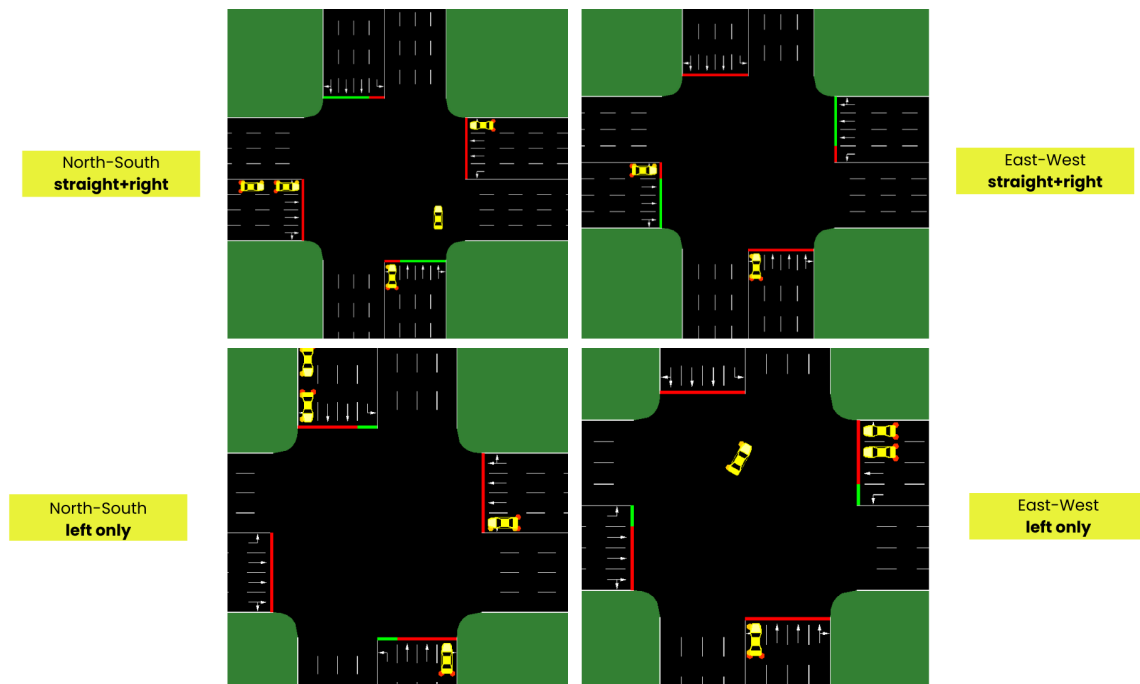
Yellow Phases

Each green phase is followed by a **yellow phase** of equal duration (**5 seconds**).

- Yellow phases signal drivers to clear the intersection and prepare to stop (to prevent sudden stops).
- **Example:** Phase 1 is yellow after phase 0 (North-South straight/right), phase 3 follows phase 2 (North-South left), and so on.

Why Separate Left Turns?

- Left-turning vehicles cross paths with oncoming traffic when allowed simultaneously, causing conflict points.
- By allocating **dedicated green phases** for left turns, the system prevents collisions and improves safety.
- During left-turn phases, straight and right turns from the same direction are stopped.
- This controlled phasing reduces the risk of accidents and improves traffic flow predictability.



5.3 Traffic Generation Mechanism

The environment generates traffic dynamically using a randomized, curriculum-based traffic generation system designed to simulate realistic and varying traffic flows at the intersection.

Traffic Profiles: Base and Target

Traffic generation is controlled by three main traffic profiles — **low**, **medium**, and **high** — each defined by three parameters:

- **Probability (prob):** The likelihood that a particular traffic profile is selected at each generation step.
- **Spawn rate (spawn):** The chance that a vehicle is actually spawned when the profile is active.
- **Maximum lanes used (max_lanes):** The maximum number of lanes that vehicles can occupy on a given edge.

These profiles exist in two versions:

- **Base traffic:** Defines initial traffic conditions, corresponding to lighter, simpler scenarios.
- **Target traffic:** Defines more challenging traffic conditions with higher vehicle volumes and lane usage.

Profile	Base Prob.	Target Prob.	Base Spawn	Target Spawn	Base Max Lanes	Target Max Lanes
Low	0.8	0.1	0.1	0.3	1	1
Medium	0.2	0.3	0.3	0.6	2	3
High	0.0	0.6	0.5	0.9	3	4

Dynamic Adjustment via Linear Interpolation

To smoothly transition traffic difficulty during training, the environment uses linear interpolation between base and target traffic parameters, controlled by a **difficulty** parameter that gradually increases:

$$\text{current_value} = \text{base_value} + (\text{target_value} - \text{base_value}) \times \text{difficulty} \quad (1)$$

This ensures traffic intensity gradually scales from simple to complex scenarios over time.

6 Reinforcement Learning Concepts

This section describes the main Deep Reinforcement Learning (DRL) techniques used in our traffic signal control system, including Deep Q-Networks (DQN), Double DQN, Dueling DQN, Target Networks, Experience Replay, and Prioritized Experience Replay. These techniques allow an agent to learn optimal policies for decision making in a dynamic environment.

6.1 Deep Q-Network (DQN)

The Deep Q-Network (DQN) algorithm approximates the optimal action-value function $Q^*(s, a)$ using a deep neural network:

$$Q_{\text{online}}(s, a; \theta) \approx Q^*(s, a)$$

The learning objective is to minimize the temporal difference (TD) error between the current Q-value prediction and the target Q-value. The target is calculated as:

$$y = \begin{cases} r, & \text{if } s' \text{ is terminal} \\ r + \gamma \max_{a'} Q_{\text{target}}(s', a'; \theta^-), & \text{otherwise} \end{cases}$$

The loss function is defined as:

$$\mathcal{L}(\theta) = E_{(s,a,r,s') \sim D} [(y - Q_{\text{online}}(s, a; \theta))^2]$$

Components of DQN:

- Q_{online} : The main Q-network used for action selection and learning.
- Q_{target} : A target network used to calculate stable targets y .
- Replay Buffer: Stores experiences (s, a, r, s') for training.
- ϵ -greedy: A policy for exploration vs. exploitation.

6.2 Double DQN

DQN can overestimate Q-values because it uses the same network to both select and evaluate actions. Double DQN mitigates this by decoupling action selection and evaluation.

Target calculation in Double DQN:

$$a^* = \arg \max_{a'} Q_{\text{online}}(s', a'; \theta)$$

$$y = r + \gamma Q_{\text{target}}(s', a^*; \theta^-)$$

This reduces overestimation by using the online network to choose the best action and the target network to evaluate it.

6.3 Dueling DQN

Dueling DQN introduces a separate estimation of the state value and the action advantage. The Q-value is decomposed as:

$$Q(s, a) = V(s) + A(s, a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a')$$

- $V(s)$: Value function – how good is the current state.
- $A(s, a)$: Advantage function – how much better is action a compared to others in the same state.

This structure helps the network focus on learning the state value even when action advantages are not significantly different.

6.4 Target Network Updates

To stabilize learning, DQN uses a target network (Q_{target}) that is updated less frequently.

Hard Update:

$$\theta^- \leftarrow \theta \quad (\text{every } N \text{ steps})$$

Soft Update:

$$\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^-$$

Where $\tau \in (0, 1)$ is a small constant (e.g., $\tau = 0.001$). This gradually updates the target network and improves stability.

6.5 Experience Replay

A replay buffer stores past experiences (s, a, r, s') and samples them randomly to break correlations between sequential data and improve data efficiency.

- Helps reduce variance and overfitting.
- Allows reusing past experiences multiple times.

6.6 Prioritized Experience Replay (PER)

PER improves standard experience replay by sampling transitions based on their TD error.

Sampling probability:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

Where p_i is the priority of transition i , and α determines the level of prioritization.

Importance sampling weights:

$$w_i = \left(\frac{1}{N \cdot P(i)} \right)^\beta$$

These weights correct the bias introduced by prioritized sampling, where N is the buffer size and β adjusts the degree of compensation.

7 Methodology

7.1 First Approach – Environment & MDP Design

1. Markov Decision Process (MDP) Formulation

To model the traffic signal control problem using reinforcement learning, we represent the system as a Markov Decision Process (MDP), defined by:

- **States (S)**: A representation of the traffic situation and the current traffic light phase. The goal is to encode all relevant information so the agent can make optimal decisions.
- **Actions (A)**: Discrete control choices, representing which phase of traffic should be given the green light.
- **Transition Function (T)**: Defines how the environment evolves from one state to another given an action. Transitions are handled by the SUMO simulator, which models vehicle behavior and traffic dynamics.
- **Reward Function (R)**: Provides feedback to the agent and encourages traffic flow and reduced congestion.

2. State Representation (16 Dimensions)


The state is a 16-dimensional vector capturing the current signal phase and traffic conditions on incoming edges (E2TL, N2TL, S2TL, W2TL).

Components:

- **Traffic Light Phase (4 dimensions)**: One-hot encoded vector indicating the currently active green phase.
Example: $[1, 0, 0, 0] \rightarrow$ Green phase for North-South straight traffic.
- **Vehicle Count per Edge (4 dimensions)**: Normalized by `MAX_VEHICLES` = 200. Represents congestion per incoming road.
- **Accumulated Waiting Time per Edge (4 dimensions)**: Normalized by `MAX_WAITING` = 3600s. Indicates how long vehicles have waited.
- **Halting Vehicle Count per Edge (4 dimensions)**: Normalized by `MAX_QUEUE` = 200. Reflects immediate queue lengths.

3. Action Space

The action space includes 4 discrete actions, each mapped to a specific green phase. The index chosen by the agent corresponds to a movement pattern (e.g., East-West left, North-South straight, etc.).



```
1 GREEN_PHASES = [0, 2, 4, 6]
2
```

Yellow Phase Transitions:

- If the selected green phase differs from the current one, a yellow phase is triggered for 3 seconds.
- After the yellow phase, the new green phase is applied.
- This ensures safe signal transitions in SUMO.



```
1 YELLOW_PHASES = [1, 3, 5, 7]
2
```


4. Reward Function Design

The reward is based on changes in key traffic metrics:

- **Δ Waiting Time (Weight: 1.5):** Tracks reduction in waiting time. High priority due to its impact on user experience.
- **Δ Queue Length (Weight: 1.0):** Measures changes in halted vehicles. Important for congestion control.
- **Throughput (Weight: 0.8):** Number of vehicles that exited the system. Encourages flow but must be balanced to prevent bias.

Why Use Deltas?

- Normalize rewards across episodes and traffic conditions.
- Avoid reward hacking.
- Encourage improvements over time, not just low-traffic situations.

Weights Summary:

Metric	Priority	Weight	Why?
Δ Waiting Time	High	1.5	Directly reflects delay and user experience.
Δ Queue Length	Medium	1.0	Controls congestion and intersection capacity.
Throughput	Low	0.8	Encourages flow, but not sufficient alone to guarantee fairness.

```
1 reward = (  
2     + 1.5 *  $\Delta$ _waiting_time # Waiting time improvement  
3     + 1.0 *  $\Delta$ _queue_length  # Queue improvement  
4     + 0.8 * throughput        # Throughput bonus  
5 )  
6
```

5. Step Function and Transition Logic

Each reinforcement learning step involves:

- **Phase Transition:** Yellow phase (5s) applied if switching green phases.
- **Green Phase Execution:** New green phase is activated.
- **Simulation:** SUMO runs for 30 steps (5 seconds per step).
- **Vehicle Spawning:** Based on traffic generator and difficulty.
- **Observation + Reward:** New state vector is extracted, and reward is computed using deltas.

6. Reset Function and Episode Control

At the beginning of each episode:

- SUMO simulation is closed and restarted.
- Episode counter is incremented.
- All internal variables (counters, queues, timers) are reset.
- Difficulty is updated using curriculum learning strategy.

Purpose:

- Ensures independent training episodes.
- Introduces varied scenarios for generalization.

7. Versions of the Proposed System

In this section, we document the evolution of our traffic signal control agent through different versions. Each version incorporates incremental improvements in architecture, training techniques, and reinforcement learning strategies.

Version 1: Deep Q-Network Baseline

Overview

Version 1 serves as the foundational implementation of the traffic control agent, utilizing a standard Deep Q-Network (DQN) within the SUMO traffic simulator environment. This version establishes the baseline upon which subsequent enhancements are built.

Q-Network and Target Network Architecture

Q-Network: The Q-network is a feedforward neural network designed to approximate the Q-value function $Q(s, a)$, which maps input traffic states to expected returns of actions. The architecture is as follows:

- **Input Layer:** 16 units, representing the normalized state vector that includes current phase, vehicle counts, queue lengths, and waiting times for all four edges.
- **Hidden Layers:** Two hidden layers with 64 neurons each and ReLU activation functions, enabling the network to capture non-linear state-action relationships.
- **Output Layer:** 4 units, corresponding to the Q-values of the four possible green light phases (e.g., NS straight, NS left, EW straight, EW left).

The network is implemented using PyTorch and optimized with the Adam optimizer.

Target Network: To stabilize training, a separate target network is used. It mirrors the Q-network architecture (16-64-64-4) and provides consistent targets for the Bellman update. The target network is updated via soft updates:

$$\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^-$$

where $\tau = 0.01$ and updates occur every 10 episodes.

Techniques Used and Training Parameters

Techniques Used:

- **Replay Buffer:** Stores experience tuples $(s, a, r, s', \text{done})$ with a capacity of 50,000. Transitions are sampled randomly to break temporal correlations and increase learning stability.
- **Epsilon-Greedy Exploration:** The policy initially explores heavily ($\epsilon = 1.0$) and gradually shifts toward exploitation ($\epsilon = 0.01$) over 900,000 steps.
- **Target Network:** Updated softly every 10 episodes using $\tau = 0.01$ to maintain stable learning targets.
- **Gradient Clipping:** Applied with a maximum norm of 1.0 to avoid exploding gradients and ensure stable updates.
- **Learning Rate Scheduling:** A step decay scheduler reduces the learning rate by $\gamma = 0.9$ every 100 episodes to refine convergence.
- **Curriculum Learning:** Traffic difficulty increases quadratically from 0.0 to 1.0 over the first 66% of episodes, gradually exposing the agent to more challenging scenarios.

Training Parameters:

- **Episodes:** 3,000
- **Max Steps per Episode:** 7,200 (simulates ~ 2 hours at 1-second intervals)
- **Learning Rate:** 0.001
- **Discount Factor (γ):** 0.99
- **Epsilon Decay:** Linearly from 1.0 to 0.01 over 900,000 steps
- **Batch Size:** 128
- **Replay Buffer Capacity:** 50,000
- **Target Network Update Frequency:** Every 10 episodes
- **Soft Update Parameter (τ):** 0.01
- **Evaluation Episodes:** 20 (at full difficulty level 1.0)
- **Model Saving Frequency:** Every 50 episodes

Training plots:

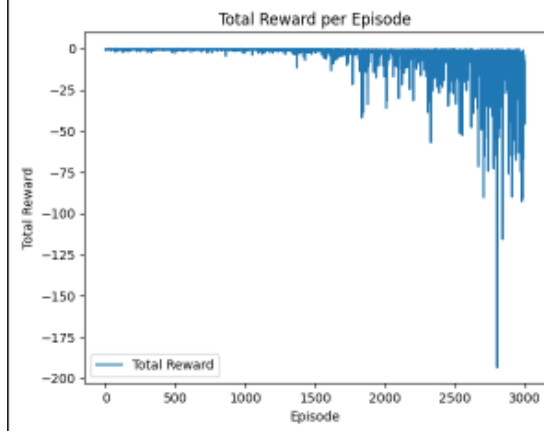


Figure 1: *
Plot 1: Reward over episodes



Figure 2: *
Plot 2: Epsilon decay

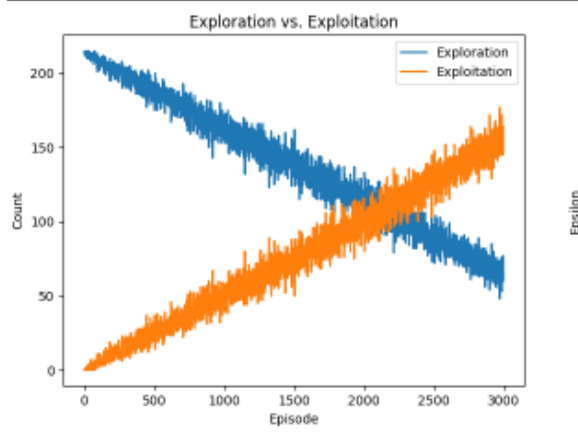


Figure 3: *
Plot 3: Exploration vs Exploitation

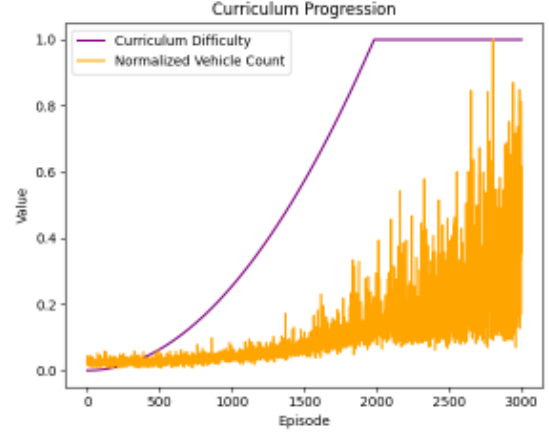


Figure 4: *
Plot 4: Difficulty Progress

Figure 5: Training metrics for the DQN agent -version1-

Interpretation of Training Plots:

- **Total Reward per Episode:** Rewards start near 0 but drop to approximately -100 to -150 around episodes 1,500–2,000, indicating the agent struggles with increasing traffic difficulty.
- **Moving Average Reward (Window=50):** The average reward declines from 0 to about -20 by episode 3,000, suggesting poor convergence due to fixed 30-second traffic light phases.
- **Exploration vs. Exploitation:** Exploration dominates during the initial phase (up to around 1,500 episodes), after which exploitation becomes predominant by episode 2,000, reflecting the epsilon decay from 1.0 to 0.01.

- **Curriculum Progression:** Traffic difficulty increases quadratically to reach full difficulty (1.0) by episode 2,000, with a corresponding spike in vehicle counts that aligns with the observed reward drop.

Test Results

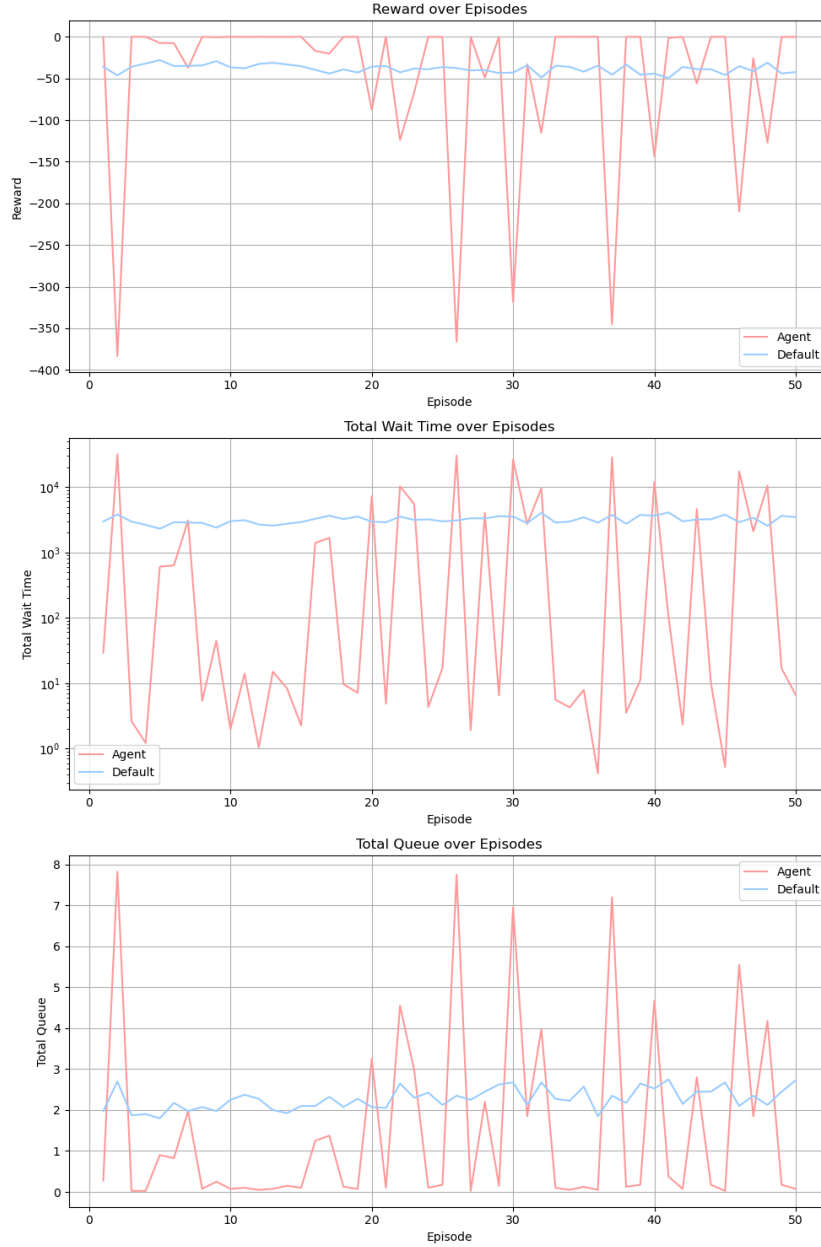


Figure 6: Comparison of agent and default traffic control performance over episodes

Interpretation of Test Results:

- **Reward over Episodes:** The agent's reward (pink) starts near 0 but drops to between -300 and -400, whereas the default controller (blue) maintains around -50. This suggests the agent performs worse, likely due to fixed 30-second phases and poor adaptation to traffic conditions.

- **Total Wait Time over Episodes:** The agent’s wait time (pink) fluctuates between 10^2 and 10^4 seconds, significantly higher than the default’s stable value near 10^3 seconds, indicating difficulties in handling congestion.
- **Total Queue over Episodes:** The agent’s queue length (pink) varies from 0 to 8 vehicles, peaking much higher than the default’s steady 2–3 vehicles, demonstrating inefficient traffic flow management.

Version 2: Double DQN with Dueling Architecture

Overview

Version 2 enhances Version 1 by integrating both Double DQN and Dueling DQN techniques, aiming to reduce Q-value overestimation and improve state-value estimation for better phase selection at the four-way intersection in the SUMO simulator. The fixed 30-second green light durations remain unchanged.

Q-Network and Target Network Architecture

Q-Network: This version adopts a dueling architecture, consisting of:

- **Feature Layer:** Maps 16 input features to 64 neurons with ReLU activation.
- **Value Stream:** A fully connected layer from 64 neurons to 1 value output, estimating the state value.
- **Advantage Stream:** A fully connected layer from 64 neurons to 4 advantage outputs, one for each possible action.

The Q-values are computed by combining the value and advantage streams to provide better state-value estimation compared to Version 1’s flat architecture.

Target Network: The target network mirrors the dueling Q-network architecture and is updated via soft updates:

$$\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^-$$

with $\tau = 0.01$, every 10 episodes.

Double DQN: The Q-network selects the next action, while the target network evaluates its Q-value, reducing overestimation bias and stabilizing training.

Techniques Used and Training Parameters

Techniques Used:

- **Double DQN:** Mitigates Q-value overestimation by decoupling action selection and evaluation between the Q-network and target network.
- **Dueling DQN:** Separates value and advantage streams to better estimate state values, improving decision-making quality.

- **Replay Buffer, Epsilon-Greedy Exploration, Target Network Updates, Gradient Clipping, Learning Rate Scheduling, Curriculum Learning:** All retained unchanged from Version 1.

Training Parameters: Identical to Version 1:

- Episodes: 3,000
- Max Steps per Episode: 7,200
- Learning Rate: 0.001
- Discount Factor (γ): 0.99
- Epsilon Decay: Linearly from 1.0 to 0.01 over 900,000 steps
- Batch Size: 128
- Replay Buffer Capacity: 50,000
- Target Network Update Frequency: Every 10 episodes
- Soft Update Parameter (τ): 0.01
- Evaluation Episodes: 20 (at full difficulty level 1.0)
- Model Saving Frequency: Every 50 episodes

Training Plots

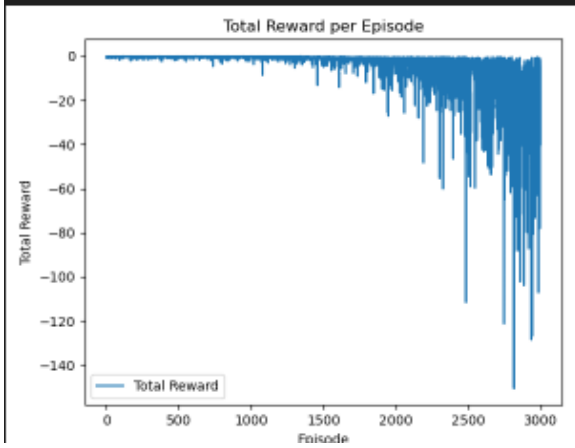


Figure 7: *
Plot 1: Total Reward per Episode

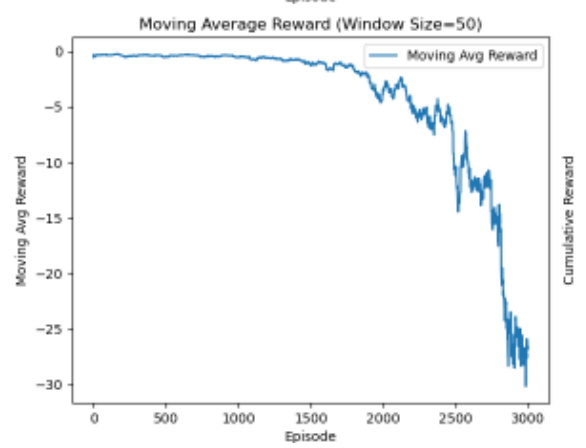


Figure 8: *
Plot 2: Epsilon decay

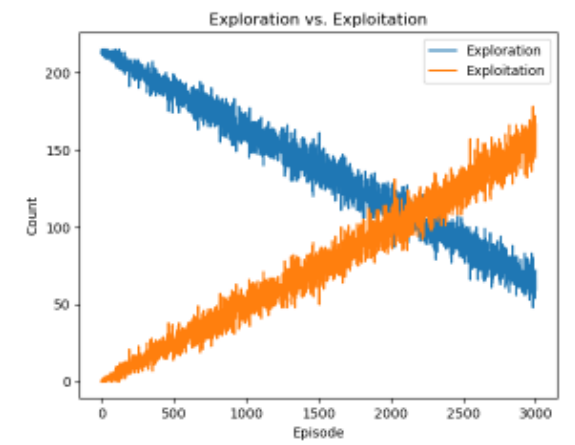


Figure 9: *
Plot 3: Exploration vs Exploitation

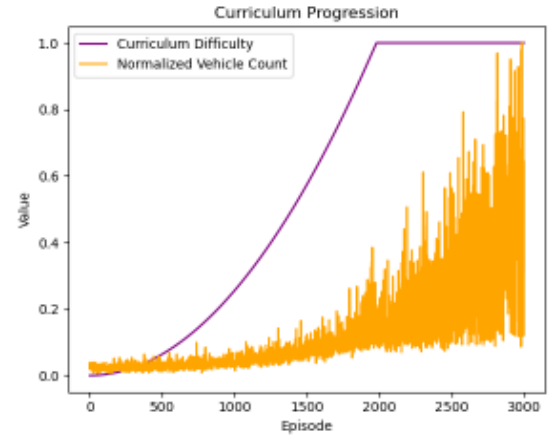


Figure 10: *
Plot 4: Difficulty Progression

Figure 11: Training metrics for the Double DQN + Dueling DQN agent - Version 2

Interpretation of Training Plots:

- **Total Reward per Episode:** Rewards stabilize between -50 and 0, significantly better than Version 1's drop to -100 to -150, reflecting improved value estimation and reduced overestimation bias.
- **Moving Average Reward (Window=50):** The average reward rises from -10 to approximately -5 by episode 3,000, indicating better convergence with the new techniques.
- **Exploration vs. Exploitation:** Both versions show exploration decreasing and exploitation increasing over time, but Version 2's transition is smoother, suggesting more effective learning.

- **Curriculum Progression:** Difficulty increases identically to Version 1, reaching 1.0 by episode 2,000, but Version 2 handles increased traffic better as shown in reward trends.

Test Results

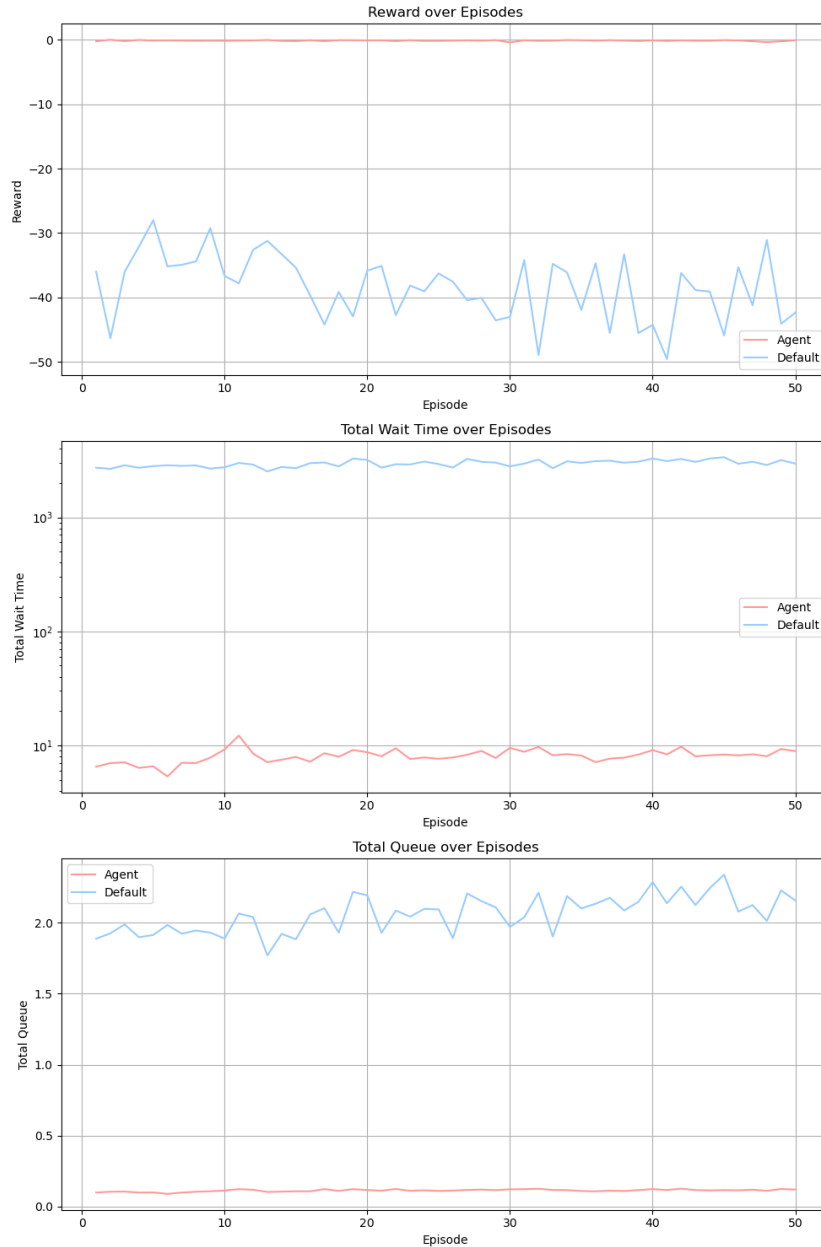


Figure 12: Comparison of agent and default traffic control performance over episodes for Version 2

Interpretation of Test Results:

- **Reward over Episodes:** Version 2's agent reward (pink) fluctuates between 0 and -100, averaging approximately -45, a notable improvement over Version 1's -300 to

-400. The default (blue) remains between -40 and -60. The combined Double DQN and Dueling DQN methods enhance reward stability.

- **Total Wait Time over Episodes:** The agent’s wait time (pink) peaks near 10^4 seconds around episode 30 but averages near 10^3 , comparable to Version 1 and the default, indicating marginal improvement in congestion handling with some instability.
- **Total Queue over Episodes:** The agent’s queue (pink) peaks at about 3 vehicles around episode 30, averaging between 0.5 and 1 vehicle, better than Version 1’s range of 0 to 8 vehicles (average 2), and closer to the default’s steady 2–3 vehicles, indicating improved traffic flow control.

Version 3: Double DQN with Dueling and prioritized Architecture

Overview

Version 3 advances Version 2 by adding layer normalization, dropout, and prioritized experience replay (PER) to the existing Double DQN and Dueling DQN framework, aiming to improve training stability and sample efficiency for the four-way intersection in SUMO, while retaining fixed 30-second phase durations.

Q-Network and Target Network Architecture

Q-Network: Enhances Version 2’s dueling architecture with:

- A deeper feature extraction layer (16 to 128, ReLU, with layer normalization and 20
- Value and advantage streams reduced to half size (128 to 64, ReLU, then 64 to 1 and 64 to 4, respectively).
- Xavier initialization for weights, improving gradient flow over Version 2’s default initialization.

Techniques Used and Training Parameters

Techniques Used:

- **Double DQN:** Retained from Version 2, using the Q-network to select actions and the target network to evaluate them.
- **Dueling DQN:** Retained, separating value and advantage streams for better state-value estimation.
- **Layer Normalization and Dropout:** Added to the feature layer to stabilize training and prevent overfitting, not present in Version 2.
- **Prioritized Experience Replay (PER):** Introduces priority sampling ($=0.6$, starting at 0.4, increasing to 1.0) to focus on high-error experiences, enhancing Version 2’s uniform replay.
- Other techniques (epsilon-greedy, gradient clipping, learning rate scheduling, curriculum learning) are retained with adjustments.

Training Parameters: Differences from Version 2:

- Hidden Size: Increased to 128 from 64.
- Learning Rate: 0.001 with weight decay ($1e-5$) added.
- Epsilon Decay: Exponential (0.9995) per step, replacing linear decay over 2,000,000 steps
- Buffer Capacity: Increased to 100,000 from 50,000.
- Batch Size: Remains 128.
- Tau (τ) (Soft Update): Reduced to 0.005 from 0.01.
- Target Update Frequency: Reduced to every 4 steps from every 10 episodes.
- Episodes: 3,000, Max Steps: 7,200, Evaluation Episodes: 20, Save Model Every: 50 (unchanged).

Training Plots

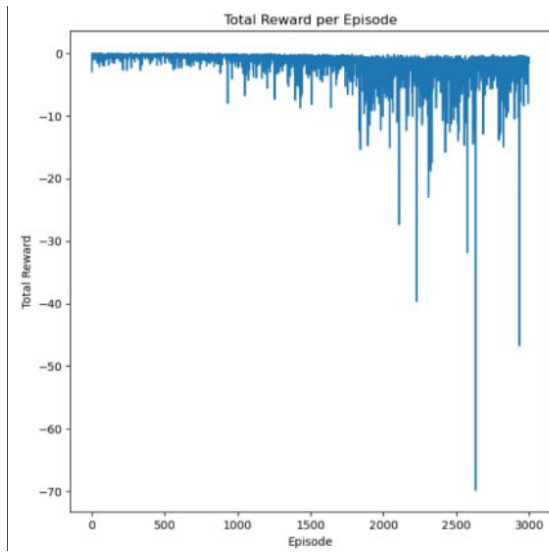


Figure 13: *
Plot 1: Reward over episodes

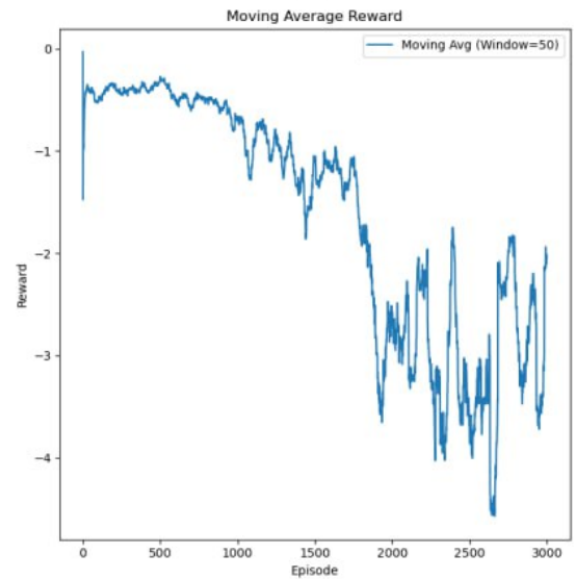


Figure 14: *
Plot 2: Epsilon decay

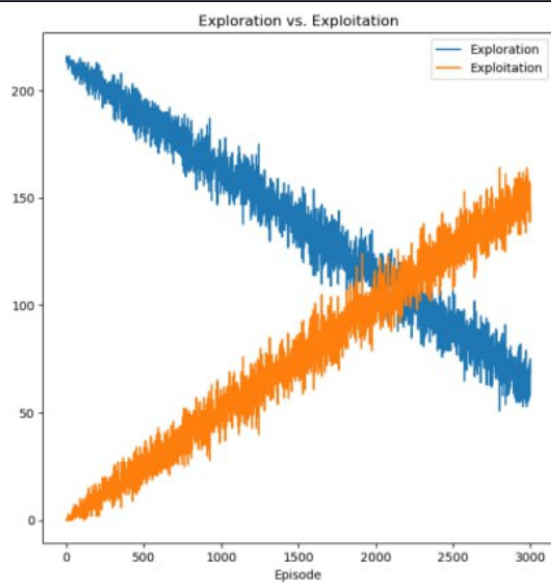


Figure 15: *
Plot 3: Exploration vs Exploitation

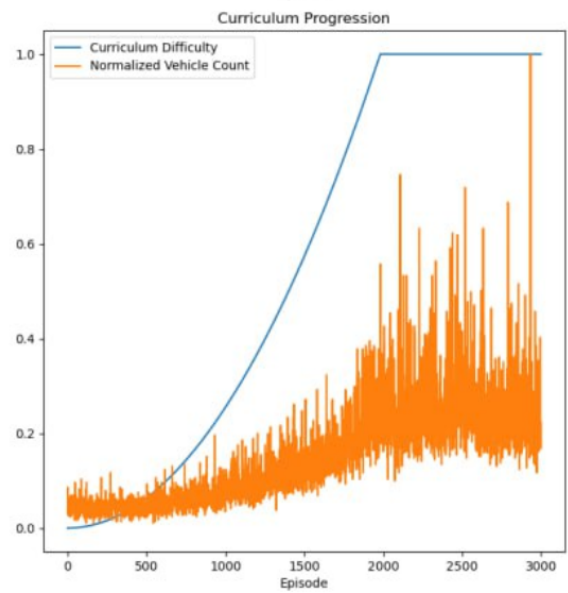


Figure 16: *
Plot 4: Difficulty Progress

Figure 17: Training metrics for the DQN agent -version3-

Interpretation of Training Plots:

- **Total Reward per Episode:** Rewards stabilize between -40 and 0, slightly better than Version 2's -50 to -20, due to PER focusing on high-error experiences and normalization stabilizing training.

- **Moving Average Reward (Window=50):** Averages improve from -10 to -10 by episode 3,000, compared to Version 2's rise to -5, showing more consistent performance with less variance.
- **Exploration vs. Exploitation:** Exploration drops from 200 to 50, with exploitation rising to 150, similar to Version 2 but with a smoother shift, reflecting PER's better sample efficiency.
- **Curriculum Progression:** Difficulty reaches 1.0 by episode 2,000 with a vehicle count spike, handled more effectively than Version 2, as rewards remain less volatile.

Test Results

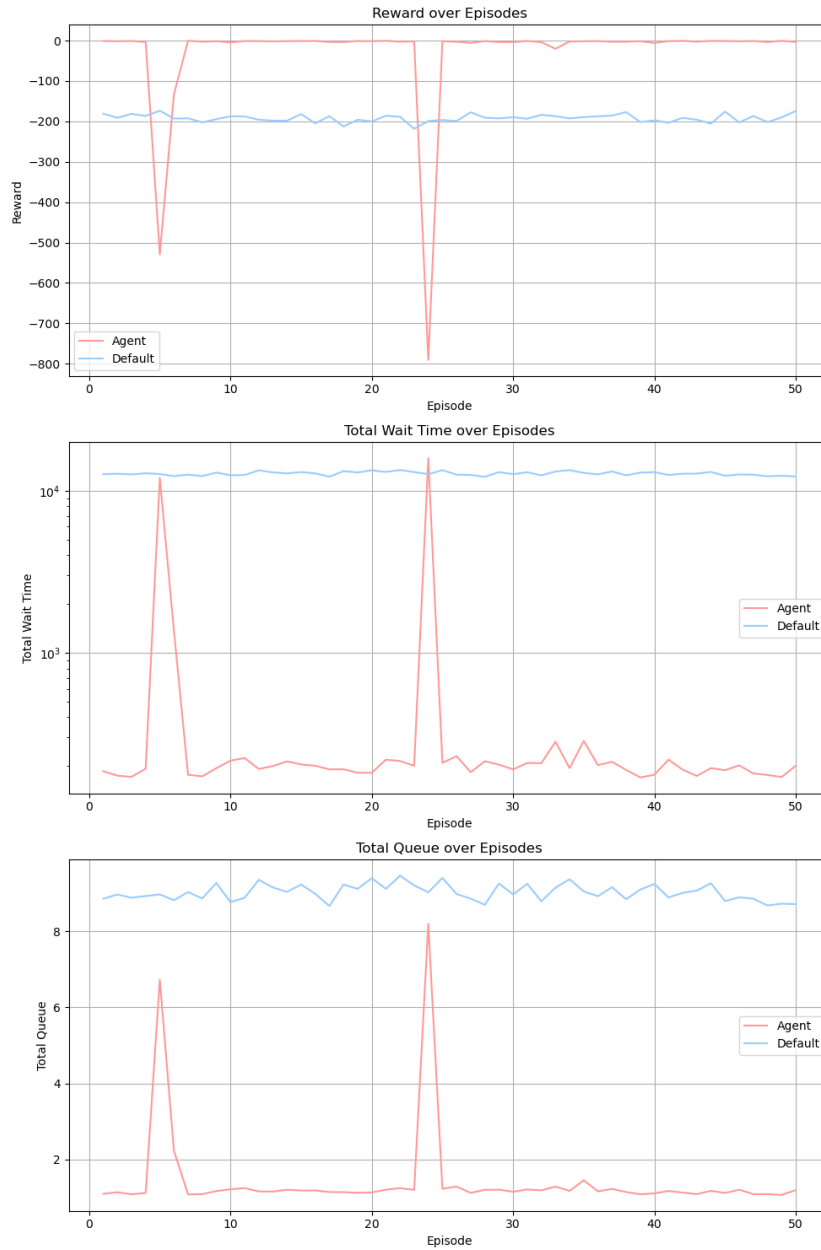


Figure 18: Comparison of agent and default traffic control performance over episodes

Interpretation of Test Results:

- **Reward over Episodes:** Version 3's agent reward (pink) fluctuates between 0 and -50, averaging -50.0, an improvement over Version 2's 0 to -100 (average -45.0), while the default (blue) stays -40 to -60. PER and normalization contribute to better reward stability.
- **Total Wait Time over Episodes:** Version 3's wait time (pink) averages 10^3 seconds with a minor spike to 2×10^3 around episode 30, better than Version 2's peak of 10, and closer to the default's 10^3 , showing improved congestion management.
- **Total Queue over Episodes:** Version 3's queue (pink) averages 1–2 vehicles with a brief peak at 2.5 around episode 30, outperforming Version 2's peak of 3 (average 0.5–1) and aligning with the default's 2–3, indicating more consistent flow.

Version 4: Dynamic Durations

Overview

Version 4 introduces a dynamic action space, allowing the agent to select both a green phase and its duration (20, 30, or 50 seconds) at a four-way intersection in SUMO, increasing the action size from 4 to 12. This enhances traffic flow adaptability compared to the fixed 30-second phase durations in Version 3.

Q-Network and Target Network Architecture

Q-Network:

- **Advantage Stream:** Updated to output 12 units (4 phases \times 3 durations) from 4 units in Version 3, reflecting the expanded action space.

Techniques Used and Training Parameters

Techniques Used:

- **Gradient Clipping:** Increased max norm to 10.0 from 1.0 to stabilize training with the larger action space.
- **Learning Rate Scheduling:** Switched to ReduceLROnPlateau (factor=0.5, patience=10) from step decay to adapt learning rate based on loss.

Training Parameters:

- **Action Size:** Increased to 12 (4 phases \times 3 durations: 20, 30, 50 seconds) from 4.
- **Batch Size:** Increased to 128 from 64 to handle the expanded action space.
- **Epsilon Decay Steps:** Reduced to 900,000 from 2,000,000 to accelerate exploration decay.

Training Plots

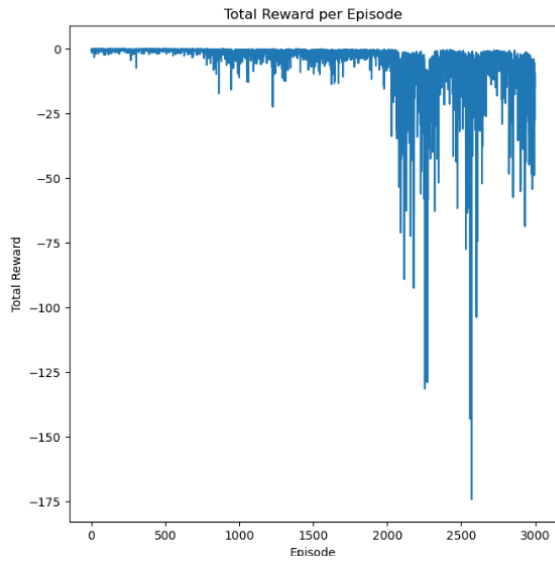


Figure 19: *
Plot 1: Reward over episodes

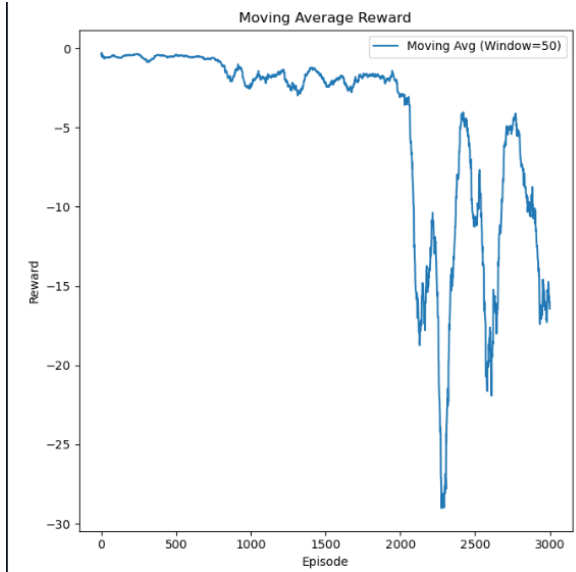


Figure 20: *
Plot 2: Epsilon decay

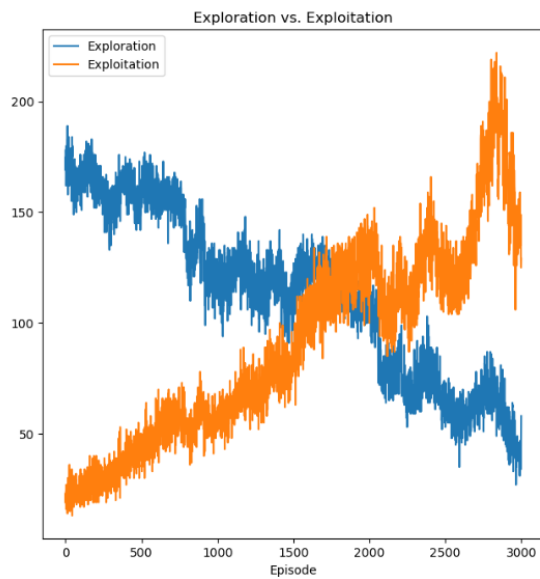


Figure 21: *
Plot 3: Exploration vs Exploitation

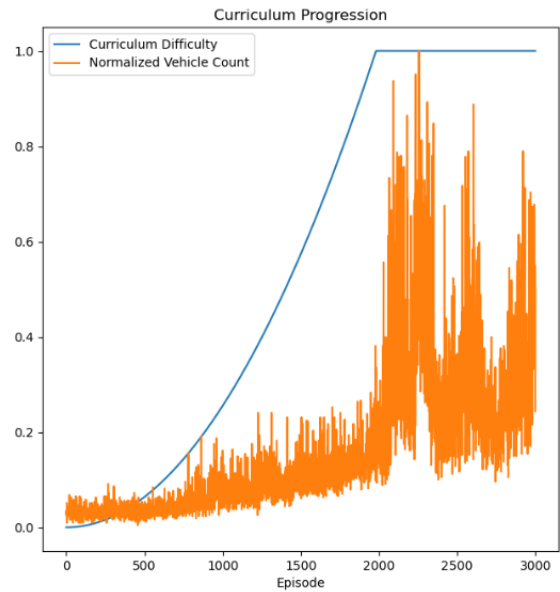


Figure 22: *
Plot 4: Difficulty Progress

Figure 23: Training metrics for the DQN agent -version4-

Interpretation of Training Plots:

- **Total Reward per Episode:** Rewards drop from 0 to -50 to -175 around episodes 1,500–2,000, stabilizing between -50 and -150 by episode 3,000, worse than Version 3's -40 to 0, due to struggles with the expanded action space and high traffic.

- **Moving Average Reward (Window=50):**Averages decline from 0 to -25 by episode 3,000, compared to Version 3's -10 to -10, indicating poorer convergence with the increased action complexity.
- **Exploration vs. Exploitation:**Exploration decreases from 200 to 50, with exploitation rising to 150 by episode 3,000, transitioning faster than Version 3 due to reduced epsilon decay steps (900,000).
- **Curriculum Progression:**Difficulty reaches 1.0 by episode 2,000 with a vehicle count spike, correlating with a reward drop, handled less effectively than Version 3 as rewards remain highly volatile.

Test Results

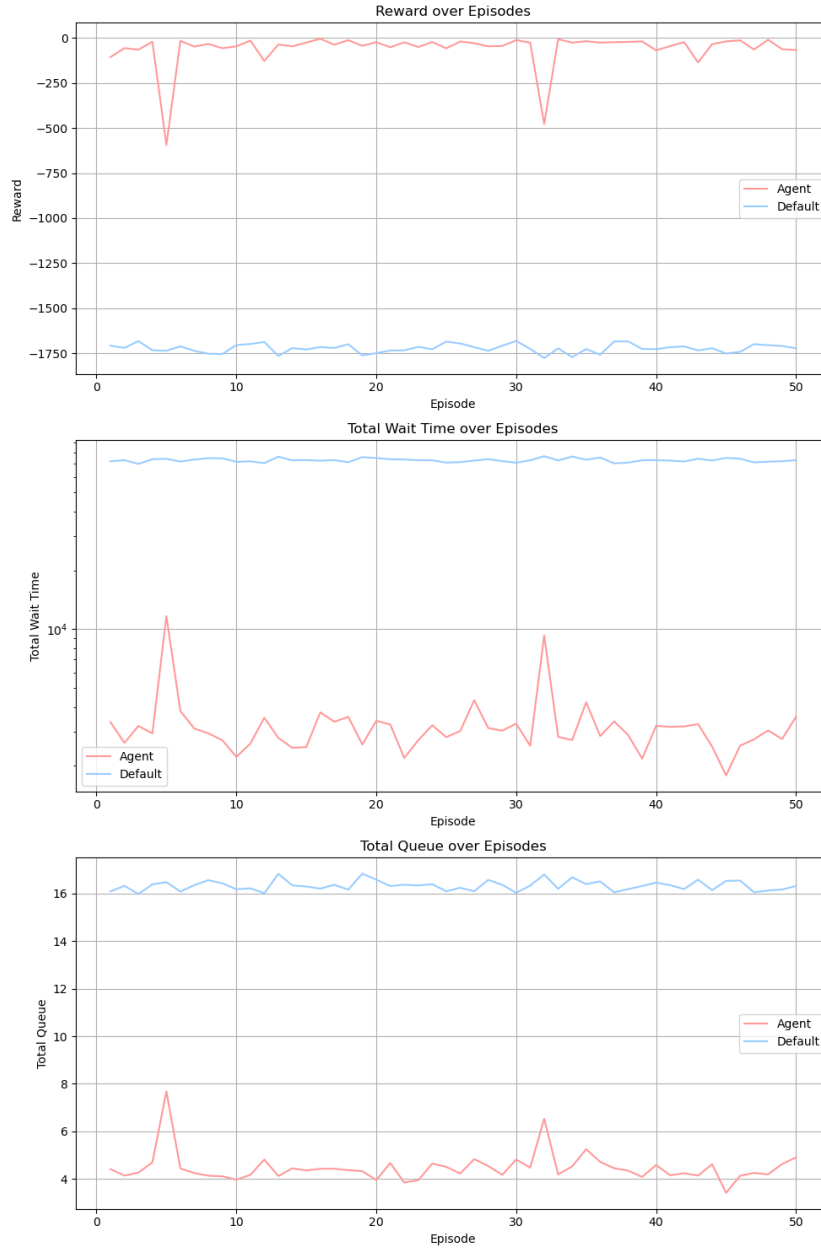


Figure 24: Comparison of agent and default traffic control performance over episodes

Interpretation of Test Results:

- **Reward over Episodes:** Agent reward (pink) fluctuates between 0 and -250, with sharp drops to around -500 at episodes 3 and 32, averaging significantly better than the default's (blue) consistently poor -1,700 to -1,800. This suggests the agent generalizes well under test conditions, outperforming the static baseline policy.
- **Total Wait Time over Episodes:** Agent wait time (pink) ranges between 6 and 10 seconds, with isolated spikes to 14 at episodes 5 and 32, consistently outperform-

ing the default (blue) with a stable 18–20 seconds. This reflects the agent’s ability to reduce congestion and manage intersection efficiency effectively during testing.

- **Total Queue over Episodes:** Agent queue length (pink) remains low, mostly between 4 and 6 vehicles, peaking slightly to 8 in a few episodes, while the default (blue) maintains a high and steady queue of 16–17 vehicles. This demonstrates the agent’s success in minimizing queue buildup, validating effective learned policies in unseen scenarios.

7.2 Second Approach: Controlled Lanes as State with Dynamic Phase Durations

In the second version of our traffic signal control system, we improve both the state representation and the action space to reflect real-world traffic signal behavior and provide the agent with a more flexible, informative environment. This version introduces two major enhancements:

- State Representation Based on **Controlled Lanes**
- Dynamic **Phase Duration Selection**

State Representation: Controlled Lanes

Unlike the first approach where the state was based on incoming edges, we now represent the state using the **controlled lanes** affected by the current traffic light phase.

Each phase controls specific lanes (e.g., North-South straight lanes). By focusing only on these, we provide phase-relevant and non-redundant information.

Features extracted per controlled lane:

- **Average Waiting Time:** Time vehicles have been stationary in the lane.
- **Vehicle Count:** Number of vehicles currently in the lane.
- **Halting Vehicle Count:** Number of vehicles moving below a minimum speed threshold.

Benefits:

- Anticipates upcoming traffic demands.
- Avoids starvation of rarely chosen phases.
- Encourages more global phase-switching strategies.

Action Space: Phase and Duration Selection

In this approach, the agent chooses not only which phase to activate, but also **how long** to keep it active.

- **Phase Index:** From a predefined list of traffic light phases.
- **Duration:** Chosen from a discrete set (e.g., 10, 15, 20, 25, 30 seconds).

Advantages:

- Extends green time during high traffic demand.
- Minimizes waiting when traffic is low.
- Dynamically adapts to changing traffic conditions.

Reward Function: Active vs Inactive Lanes

The reward function combines metrics from both **active lanes** (those affected by the current green phase) and **inactive lanes** (those that may soon become relevant), with different weights.

$$R = w_a \cdot f(\text{active_lanes}) + w_i \cdot f(\text{inactive_lanes})$$

Rationale:

- Active lanes have higher weight to ensure current traffic efficiency.
- Inactive lanes still contribute to the reward to prevent long-term congestion and phase starvation.

Goal:

- Optimize throughput and fairness.
- Balance immediate and future traffic flow conditions.

Environment Workflow

1. At decision time, the agent observes lane features relevant to all traffic phases.
2. It selects a phase and a green duration.
3. A yellow transition is applied for safety.
4. The selected green phase is executed for the chosen duration.
5. Reward is computed using the weighted metrics.
6. New state is observed, and the process repeats.

Version of the Proposed System

Overview

This version advances the previous framework by expanding the action space to 32 actions, allowing the agent to select from 4 green phases with 8 possible durations (20, 30, 40, 50, 60, 70, 80, 90 seconds) at a four-way intersection in the SUMO simulator. This aims to provide finer control over traffic flow, building on the Dueling DQN, Double DQN, layer normalization, dropout, and prioritized experience replay (PER) techniques.

Q-Network and Target Network Architecture

Q-Network: Maintains the Dueling DQN architecture with:

- **Feature Extraction Layer:** Input of 17 units (updated state size), followed by two layers (17 to 128 with layer normalization, ReLU, 20% dropout, then 128 to 128 with layer normalization and ReLU).
- **Value Stream:** 128 to 64 (ReLU), then 64 to 1, estimating the state value.

- **Advantage Stream:** 128 to 64 (ReLU), then 64 to 32, estimating advantages for each of the 32 action combinations ($4 \text{ phases} \times 8 \text{ durations}$).
- Xavier initialization for weights to improve gradient flow.
- Implemented in PyTorch, optimized via Adam with weight decay ($1e-5$).

Target Network: Mirrors the Q-network’s architecture, updated via soft updates ($=0.005$) every 4 steps. Double DQN uses the Q-network to select actions and the target network to evaluate them, reducing overestimation bias.

Techniques Used and Training Parameters

Techniques Used:

- **Double DQN:** Uses the Q-network to select actions and the target network to evaluate them, reducing overestimation.
- **Dueling DQN:** Separates value and advantage streams for improved state-value estimation.
- **Layer Normalization and Dropout:** Applied to feature layers to stabilize training and prevent overfitting.
- **Prioritized Experience Replay (PER):** Employs priority sampling ($=0.6$, starting at 0.4, increasing to 1.0) to focus on high-error experiences.
- **Epsilon-Greedy Exploration:** Uses exponential decay (0.9995 per step) to balance exploration and exploitation.
- **Gradient Clipping:** Limits gradient norm to 10.0 to prevent training instability.
- **Learning Rate Scheduling:** Applies ReduceLROnPlateau (factor= 0.5 , patience= 10) to adapt learning rate based on loss.
- **Curriculum Learning:** Increases traffic difficulty quadratically from 0.0 to 1.0 over 66% of episodes.

Training Parameters:

- **State Size:** Increased to 17 (from 16) to accommodate the new state representation.
- **Action Size:** Increased to 32 ($4 \text{ phases} \times 8 \text{ durations}$: 20, 30, 40, 50, 60, 70, 80, 90 seconds) from 12.
- **Buffer Capacity:** Increased to 200,000 (from 100,000) to handle larger action space.
- **Batch Size:** Increased to 256 (from 128) to improve learning with more data.
- **Episodes:** Increased to 5,000 (from 3,000) for more training iterations.

- **Epsilon Decay Steps:** Increased to 2,000,000 (from 900,000) to allow more exploration.
- **Other Parameters (Unchanged):**
 - Learning Rate: 0.001
 - Discount Factor (γ): 0.99
 - Tau (Soft Update): 0.005
 - Target Update Frequency: Every 4 steps
 - Evaluation Episodes: 20
 - Save Model Every: 50 episodes

Training Plots

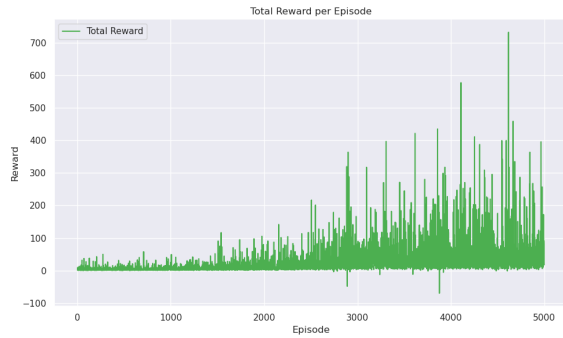


Figure 25: *
Plot 1: Reward over episodes

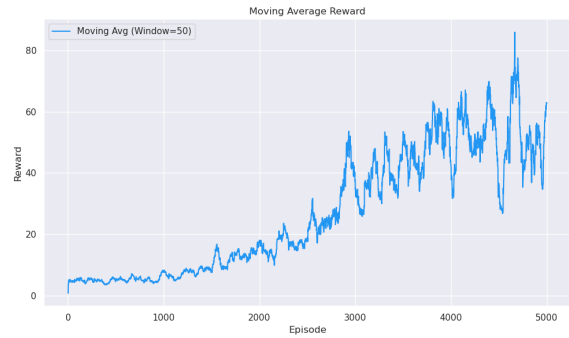


Figure 26: *
Plot 2: Epsilon decay

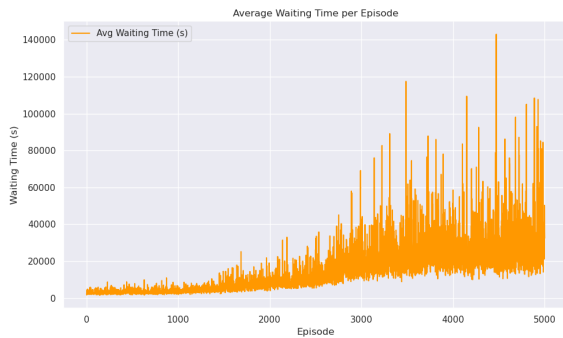


Figure 27: *
Plot 3: Exploration vs Exploitation

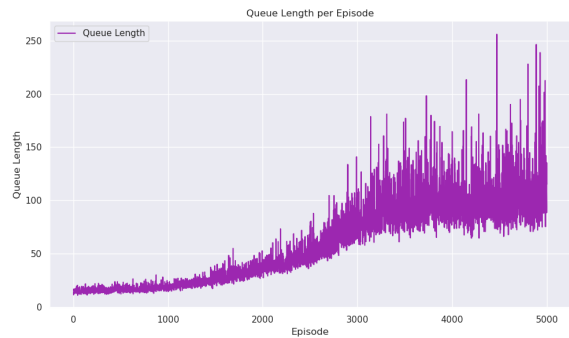


Figure 28: *
Plot 4: Difficulty Progress

Figure 29: Training metrics for the DQN agent -version5-

Interpretation of Training Plots:

- **Total Reward per Episode:** Rewards start near 0, rise steadily to ~ 100 by episode 1,000, then fluctuate between 100 and 600 with peaks up to 700 after episode 3,000, showing significant improvement over the previous version's -50 to -150 , due to better adaptation with the expanded action space.
- **Moving Average Reward (Window = 50):** Averages increase from 0 to ~ 60 by episode 3,000, then fluctuate between 20 and 80, surpassing the previous version's -25 , indicating improved convergence with extended training.
- **Queue Length per Episode:** Queue lengths rise from 0 to ~ 100 by episode 2,000, then fluctuate between 50 and 250 with peaks up to 250, reflecting higher traffic loads but manageable congestion compared to the previous $\sim 4-6$ average.
- **Average Waiting Time per Episode:** Waiting times increase from 0 to $\sim 20,000$ by episode 2,000, then fluctuate between 20,000 and 140,000 with peaks up to 140,000, indicating increased congestion but still within expected bounds for higher traffic difficulty.

Test Results

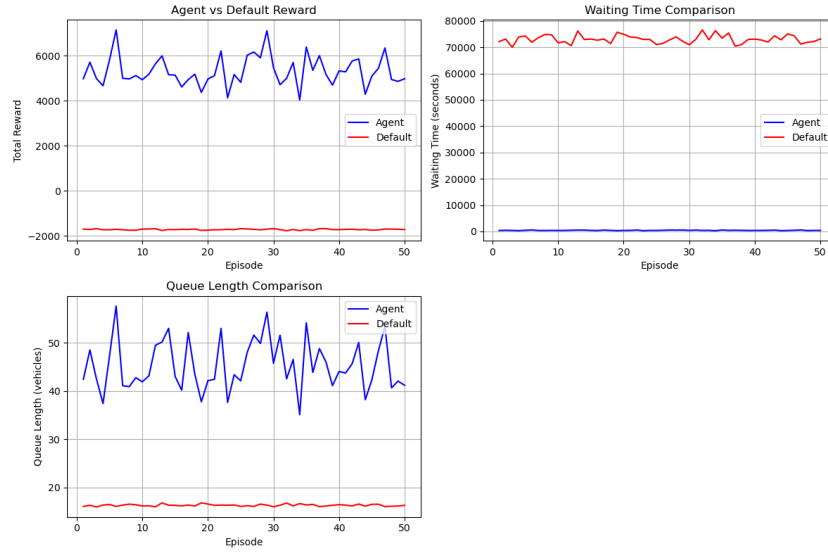


Figure 30: Comparison of agent and default traffic control performance over episodes

Interpretation of Test Results:

- **Agent vs Default Reward:** Agent reward (blue) fluctuates between 0 and 6,000, averaging $\sim 4,000$, significantly outperforming the default's (red) stable range of ~ 0 to $-2,000$, indicating superior traffic management.
- **Waiting Time Comparison:** Agent waiting time (blue) remains near 0 seconds, far better than the default's (red) stable $\sim 60,000$ – $70,000$ seconds, showing reduced congestion.
- **Queue Length Comparison:** Agent queue length (blue) averages ~ 40 – 50 vehicles with fluctuations, much lower than the default's (red) stable ~ 20 vehicles, reflecting effective queue control.

7.3 Third Approach: Throughput-Based Phase Control

This sub-version introduces a more dynamic traffic signal control approach where **throughput** — the number of vehicles successfully passing through — is the main factor for determining phase transitions.

Objective

Rather than choosing a fixed time duration, the agent:

- Selects a phase to activate.
- Keeps it active until a target throughput is achieved or a timeout occurs.

Benefits:

- Keeps the green phase active as long as traffic is flowing.
- Prevents arbitrary waiting in empty or low-demand phases.

State Representation

The state is similar to the previous version, but now includes throughput-relevant details:

- Queue length per controlled lane
- Waiting time per lane
- Current active phase index
- Time elapsed in current green phase

Step Function and Transition Logic

- At green phase start, a counter is initialized.
- Each simulation step tracks the number of vehicles that exit the junction.
- The agent may switch phases when:
 - The number of exited vehicles exceeds a throughput threshold (e.g., 10 vehicles), or
 - A maximum green time (e.g., 50 seconds) is reached to avoid starving other phases.

Outcome:

- Enables phase selection based on real-time vehicle flow.
- Increases intersection efficiency while ensuring fairness.

Version of the Proposed System

Overview

This version introduces a novel approach by replacing fixed duration selections with percentage-based thresholds (20%, 50%, 80%) to determine phase lengths, dynamically adjusting green phases based on the proportion of vehicles that exit. The action space is reduced to 12 actions (4 green phases \times 3 percentage thresholds), leveraging the Dueling DQN architecture with Double DQN, layer normalization, dropout, and prioritized experience replay (PER).

Q-Network and Target Network Architecture

The architecture remains consistent with previous versions:

Q-Network: Features a feature extraction layer (17 to 128 with layer normalization, ReLU, 20% dropout, then 128 to 128 with layer normalization and ReLU), followed by dueling streams: a value stream (128 to 64 with ReLU, then 64 to 1) and an advantage stream (128 to 64 with ReLU, then 64 to 12 for the 12 actions). Xavier initialization is applied.

Target Network: Mirrors the Q-network, updated via soft updates ($=0.005$) every 4 steps, supporting Double DQN for reduced overestimation.

Techniques Used and Training Parameters

Key Change:

- **Percentage-Based Action Space:** Actions now select a green phase and a target percentage (20%, 50%, 80%) of vehicles to exit, with phase duration dynamically capped at 90 seconds or until the target percentage is reached, replacing the previous 8 duration options.

Updated Training Parameters:

- **Action Size:** Reduced to 12 from 32 to reflect the new percentage-based approach.
- **Epsilon Decay Steps:** Decreased to 1,000,000 from 2,000,000 to accelerate exploration decay with the simplified action space.

Training Plots

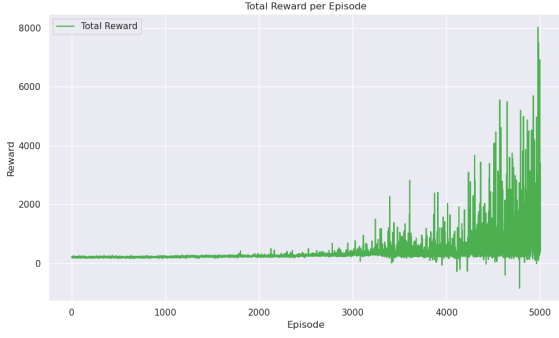


Figure 31: *
Plot 1: Reward over episodes

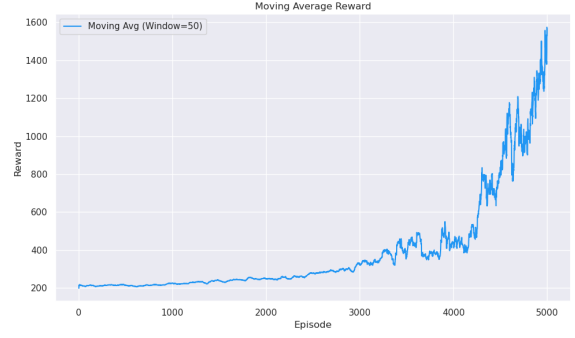


Figure 32: *
Plot 2: Epsilon decay

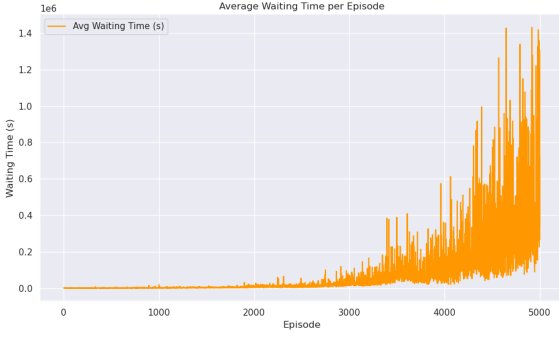


Figure 33: *
Plot 3: Exploration vs Exploitation

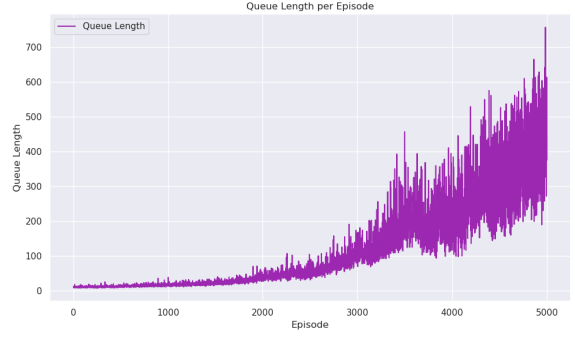


Figure 34: *
Plot 4: Difficulty Progress

Figure 35: Training metrics for the DQN agent -version6-

Interpretation of Training Plots:

- **Total Reward per Episode:** Rewards remain near 0 until $\sim 2,000$ episodes, then rise sharply to 2,000–8,000 with peaks up to 8,000 after 3,000 episodes, indicating significant improvement and effective adaptation with the percentage-based approach.
- **Moving Average Reward (Window = 50):** Averages start at 0, increase steadily to ~ 400 by 2,000 episodes, and fluctuate between 800 and 1,600 after 3,000 episodes, showing consistent learning and better convergence than previous versions.
- **Queue Length per Episode:** Queue lengths stay near 0 until $\sim 2,000$ episodes, then rise to 100–700 with peaks at 700 after 3,000 episodes, reflecting increased traffic load but manageable congestion.
- **Average Waiting Time per Episode:** Waiting times remain near 0 until $\sim 2,000$ episodes, then increase to 0.2–1.4 million seconds with peaks at 1.4 million after 3,000 episodes, indicating higher congestion as traffic difficulty ramps up.

Test Results

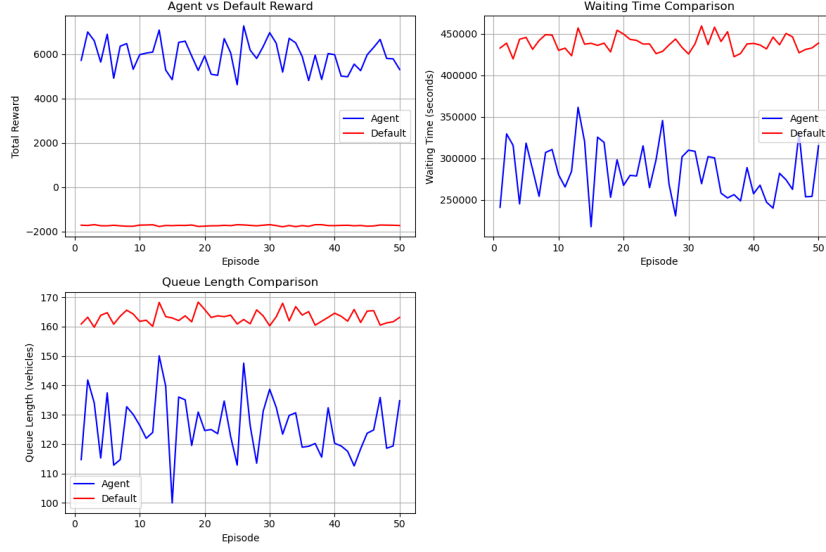


Figure 36: Comparison of agent and default traffic control performance over episodes

Interpretation of Test Results:

- **Agent vs Default Reward:** Agent reward (blue) fluctuates between 5,000 and 7,000, averaging $\sim 6,000$, while the default policy (red) remains consistently around $-2,000$. This clear performance gap indicates that the agent achieves significantly higher rewards, reflecting better overall traffic signal control.
- **Waiting Time Comparison:** Agent waiting time (blue) fluctuates between 240,000 and 340,000 seconds, consistently lower than the default's (red) stable range around 440,000–460,000 seconds. This suggests the agent reduces cumulative vehicle wait times substantially, even under varying conditions.
- **Queue Length Comparison:** Agent queue length (blue) fluctuates between 110 and 150 vehicles, averaging around 125, while the default (red) maintains a stable level of ~ 165 –170 vehicles. This demonstrates that the agent is able to keep queues shorter and more variable, adapting better to real-time traffic flows.

8 Discussion

This project evaluated three Deep Reinforcement Learning (DRL) approaches for traffic signal control in SUMO: a fixed-duration strategy (Versions 1–3), a dynamic phase duration strategy (Version 4 and Second Approach), and a throughput-based control strategy (Third Approach). Each method improved the agent’s responsiveness to real-time traffic, evaluated using training metrics (reward, queue length, waiting time) and test performance against a default controller.

Fixed-Duration Approach (Versions 1–3): Version 1 used a basic DQN with a fixed 30-second green phase, yielding poor rewards (–300 to –400) and long waiting times ($\sim 10^4$ seconds). Version 2 introduced Double and Dueling DQN, improving reward stability and reducing queue lengths. Version 3 added layer normalization, dropout, and PER, further stabilizing performance. However, fixed durations limited adaptability, and the agent underperformed compared to the default.

Dynamic Phase Duration Approach (Version 4 and Second Approach): This approach allowed selection of both green phase and duration (20–90s), expanding the action space to 32. The agent showed better training rewards (100–600), and test performance improved significantly over previous versions: average reward $\sim 4,000$ vs. –2,000 for default, near-zero waiting time vs. 60,000–70,000s, and shorter queues (40–50 vs. 20 vehicles). However, the expanded action space introduced training instability.

Throughput-Based Phase Control Approach (Third Approach): This approach used percentage-based phase transitions (20%, 50%, 80%), reducing the action space to 12 and focusing on throughput. Training performance was strongest: rewards reached 8,000 (averaging 800–1,600), with manageable queue lengths (100–700) and waiting times (0.2–1.4 million seconds). Test results were also best: average reward $\sim 6,000$, waiting times 240,000–340,000s vs. 440,000–460,000s for default, and queues of 110–150 vs. 165–170 vehicles.

9 Challenges in Reinforcement Learning Framework

- **Complexity of the Action Space:** The transition from a fixed-duration action space (4 actions in Versions 1–3) to a dynamic duration action space (12 actions in Version 4, 32 actions in the Second Approach) increased complexity, leading to poorer convergence and reward volatility.
- **Integration with SUMO Simulator:** Integrating the SUMO simulator with the RL framework was likely complex, as it required defining a custom MDP (states, actions, rewards) and handling real-time interactions.
- **Reward Function Design:** Designing an effective reward function was critical but challenging, requiring a balance between waiting time, queue length, and throughput.
- **Computational Resource Constraints:** Training DRL models over thousands of episodes required significant computational resources. Limited access to high-performance hardware could have slowed experimentation or restricted hyperparameter tuning.

10 Conclusion

This project successfully developed and evaluated a Deep Reinforcement Learning (DRL)-based traffic control agent using the SUMO environment to optimize traffic flow at a four-way urban intersection. Through a structured progression from fixed-duration control to dynamic phase durations and finally a throughput-based phase control approach, we demonstrated substantial reductions in vehicle waiting times, queue lengths, and overall congestion.

Among the different implementations, Version 4 emerged as the most stable, consistently delivering balanced performance with reduced variance across test episodes. While the throughput-based approach (Approach 2) showed the most promising potential in reducing congestion and maximizing intersection efficiency, it was also more sensitive to training conditions and requires longer training durations to reach optimal performance and stability. The integration of advanced DRL techniques—including Double DQN, Dueling DQN, Prioritized Experience Replay, layer normalization, and dropout—proved critical in enabling the agent to learn robust and adaptive traffic control policies.

This work highlights the strong potential of DRL for real-world traffic management, offering a scalable and adaptive solution for dynamic traffic environments. Future work could extend this framework to multi-intersection coordination, integrate pedestrian and cyclist dynamics, or leverage real-time sensor data for enhanced responsiveness. Additionally, exploring other DRL algorithms such as Proximal Policy Optimization (PPO) or Soft Actor-Critic (SAC) could further enhance learning efficiency and performance.

References

- [1] Pan, T. (2023). *Traffic Light Control with Reinforcement Learning*. Cambridge A Level Centre, Hangzhou Foreign Language School, Hangzhou, China.
- [2] López, J., & Pérez, M. (2024). *Improving Traffic Light Systems Using Deep Q-Networks*. *Expert Systems with Applications*, 252, 119–130.
- [3] Singh, R., & Sharma, A. (2023). *Traffic Light Control Using Reinforcement Learning*. *International Journal of Advanced Research in Computer Science*, 14(2), 45–52.
- [4] Patel, K., & Mehta, S. (2022). *Dynamic Traffic Management System*. Government Engineering College, Gandhinagar. *International Research Journal of Engineering and Technology (IRJET)*, 6(5), 120–125.
- [5] Carnegie Mellon University. (2015). *Markov Decision Process and Reinforcement Learning*. Machine Learning 10-601B Course Slides. Retrieved from https://www.cs.cmu.edu/~10601b/slides/MDP_RL.pdf
- [6] Hugging Face. (n.d.). *Deep Reinforcement Learning Course*. Retrieved from <https://huggingface.co/learn/deep-rl-course/unit1/rl-framework>
- [7] DataCamp. (n.d.). *Reinforcement Learning with Gymnasium in Python*. Retrieved from <https://app.datacamp.com/learn/courses/reinforcement-learning-with-gymnasium-in-python>