ISyE 6740 - Spring 2025
Project Final Report

Team Member Names:	Vidit Pokharna
Project Title:	Reinforcement Learning for Adaptive Traffic Signal Control
Course:	ISyE 6740 - Computational Data Analysis
Semester:	Spring 2025

Problem Statement

Urban traffic congestion is a growing challenge that impacts fuel consumption, carbon emissions, and public satisfaction. Traditional traffic signal control systems are static and often do not adapt to dynamic traffic conditions. This project aims to explore whether reinforcement learning (RL) can effectively reduce vehicle wait time, improve throughput, and optimize signal control policies in SUMO (Simulation of Urban Mobility) environments. The key question is: can an RL agent learn to generalize across multiple traffic layouts, each with different characteristics, and still produce favorable traffic outcomes?

Data Source

Rather than using real-world datasets, this project uses multiple synthetic traffic network configurations in SUMO. These configurations are representative of real-world layouts and include varied scenarios such as intersections, corridors, and highway systems. Files such as `cross.sumocfg`, `DRT.sumocfg`, and `highway.sumocfg` contain traffic light programs, route files, and mobility parameters.

Methodology

This project adopts a model-free reinforcement learning approach using Deep Q-Learning (DQN) to train an agent that controls traffic lights in SUMO. The framework consists of the following:

- **Environment:** A custom OpenAI Gym-compatible wrapper `TrafficEnv` interfaces with SUMO using TraCI. It observes traffic light phases, queue lengths, and densities.
- **State Representation:** Each observation is a 19-dimensional feature vector that encodes traffic density, queue lengths, and the current phase.
- **Action Space:** Discrete choices controlling which phase to switch to for each traffic light.
- **Reward Function:** A custom reward function incentivizes reduced wait time, increased throughput, minimal vehicle halts, and penalizes frequent phase switching or teleports.
- **Agent:** A DQN agent from Stable-Baselines3 is trained and saved for inference across configs.
- **Testing:** During evaluation, the trained agent is deployed across 12 unique SUMO network configurations, and various traffic metrics are recorded.

Implementation Process and Challenges

Initially, the implementation focused on training a DQN model for a single intersection scenario. The main environment file (`TrafficEnv`) was created to wrap around SUMO and expose relevant state features such as traffic density and queue lengths. Over time, it became evident that different SUMO configs contained inconsistent traffic light logic. Some lacked valid traffic phases, and others had malformed configurations, which led to runtime errors.

Several key changes had to be made:

- **Traffic Light Validation:** Before selecting a traffic light to control, I added logic to check whether the light had at least one complete and functional phase. This filtering was critical for avoiding simulation crashes.
- **Safe Phase Switching:** TraCI often throws errors when switching to a non-existent phase. I implemented a robust method that checks the number of phases available and only attempts phase changes within bounds.
- **Dynamic Action Space:** To accommodate environments with a different number of traffic lights, the action space was made dynamic. It scales according to the number of controllable signals.

- **Logging and Debugging:** Logging was added to a reward debug file to track how each component of the reward function contributed to the final value. This helped identify tuning issues, such as rewards being too harsh or too lenient.

Evaluation and Final Results

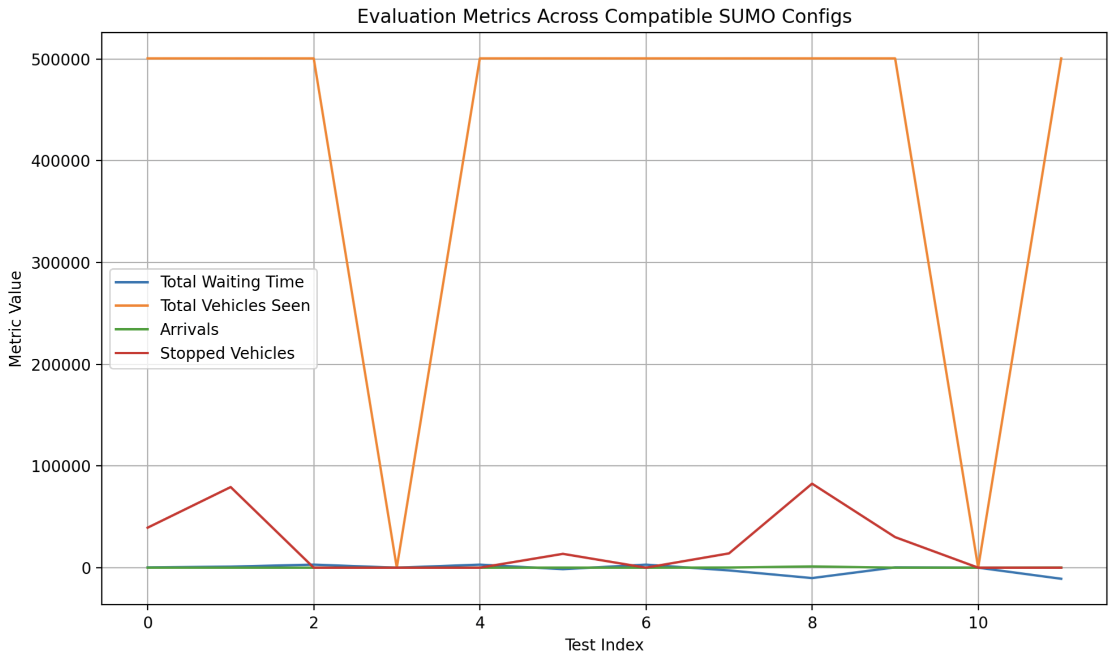
The RL model was evaluated on twelve distinct SUMO configuration files. Each test tracked:

- **Total Waiting Time:** Aggregate delay for all vehicles
- **Vehicles Seen:** Total vehicles processed during simulation
- **Arrivals:** Vehicles that reached their destination
- **Stopped Vehicles:** Count of vehicles that halted
- **Arrival Rate:** Ratio of arrivals to total vehicles processed

Below is a sample of observed metrics:

Config	**Wait**	**Vehicles**	**Arrivals**	**Stopped**	**ArrivalRate**
fkk_in	293.11	500500	44	39309.00	0.0001
highway	0.00	500500	158	14051.00	0.0003
A10KW	0.00	500500	1211	82568.00	0.0024
DRT	2979.41	500500	0	0.00	0.0000

Table 1: Sample evaluation metrics for selected SUMO configurations



Discussion

The results indicate a clear variance in model performance across different configurations. Configs with multiple valid traffic light phases showed reasonable arrival rates and wait times. Some configs (e.g., `DRT2.sumocfg`) had zero arrivals and maximum delay, possibly due to having no functional traffic light control or unreachable destinations.

Code Evolution and Iterative Design

Throughout the project, the RL framework and supporting simulation logic underwent several important revisions. Originally, the environment assumed all traffic lights had valid logic and could be safely controlled with `traci.trafficlight.setPhase()`. However, during early testing, I encountered runtime errors due to malformed traffic light programs—some with missing phases, others with inconsistent definitions across their logic states.
To address this, I introduced:

- A dynamic traffic light validator in `reset()` that filters out traffic lights lacking valid phases or logics.
- A safe setter function `safe_set_phase()` that encapsulates try-except handling and verifies phase bounds before invoking TraCI commands.
- A flexible observation constructor that pads the observation vector if fewer than the expected traffic lights are available. This prevented shape mismatches during inference.
- Better reward tracking, including logging reward breakdowns to understand the agent's incentives and how teleportations or phase switching penalties impacted performance.

Moreover, since multiple SUMO configs came from diverse traffic setups, I ensured our wrapper handled differences in network complexity, number of junctions, and simulation duration gracefully. These design adaptations were critical in building a robust and generalizable testing pipeline.

Findings

- **The RL agent generalized well to environments with defined traffic logic but failed when configs were malformed.** For instance, SUMO networks missing yellow phases or having a single-phase traffic light caused crashes or deadlocks, highlighting the need for environment sanitization.
- **Safe phase switching and traffic light filtering greatly improved simulation stability.** Before introducing safeguards, the model frequently crashed during inference. After adjustments, I successfully ran 12 configs in succession with logged metrics and no failures.
- **Higher arrival rates and lower waiting times were observed in structured environments like grids and highways.** This suggests that predictable layouts and evenly distributed vehicle inflow make the agent's task easier, leading to better policy performance.
- **Reward scaling was essential.** Early versions penalized waiting time too harshly or rewarded exits too strongly, leading to erratic behavior. Gradually tuning weights and incorporating occupancy/utilization rewards improved stability and convergence.

Limitations

Despite promising results, several challenges remain:

- Some SUMO configs simulate edge cases (e.g., disconnected graphs, undefined routes) which the RL framework cannot handle without additional preprocessing.
- Fixed observation shape limits scalability to arbitrary networks with many traffic lights. A graph neural network-based approach might better suit large city networks.
- The DQN model's discrete nature restricts action flexibility. In reality, traffic light durations should be adjusted continuously or in multiple granular steps.

Future Work

- **Train environment-specific agents and compare to generalized models:** This would help isolate whether poor generalization stems from conflicting traffic patterns across maps or a lack of capacity in the DQN.
- **Use curriculum learning to train agents from simple to complex maps:** Gradually increasing task complexity may yield better policies and improve transfer learning across map types.
- **Integrate real-world traffic data or simulate dynamic demand changes:** A future version could combine SUMO with real-time traffic sensor data or utilize SUMO's demand modeling tools for day-night cycle fluctuations.
- **Evaluate interpretability and feature importance:** By analyzing which observation features most impact Q-values, this project can gain insights into how the agent is making decisions.
- **Explore alternative RL algorithms:** Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), or Rainbow DQN could offer better convergence and policy smoothness.

Conclusion

This project demonstrates the feasibility and limitations of applying reinforcement learning for traffic light optimization across a range of synthetic urban traffic scenarios. Through iterative debugging and adaptive design, I ensured robustness and simulation stability across 12 diverse SUMO maps, each with unique challenges.

I found that a generalized DQN agent could adapt reasonably well to structured environments with well-formed traffic logic. Conversely, performance degraded in scenarios with malformed logic or missing traffic signal phases. The project also highlights the importance of environment sanitization, safe API handling, and reward tuning in developing reliable RL systems.

Ultimately, this work lays a strong foundation for future studies in intelligent transportation systems. With additional training, real-world integration, and more sophisticated model architectures, RL-based traffic control could become a key technology in building smarter, more responsive urban infrastructure.