République algérienne démocratique et populaire

الجمهورية الجزائرية الديمقراطية.

Ministère de l'enseignement supérieur et de la recherche scientifique

وزارة التعليم العالي والبحث العلمي.

École supérieure en Sciences et Technologies de l'Informatique et du Numérique

المدرسة العليا في علوم وتكنولوجيات الإعلام الآلي والرقمنة.

Reinforcement Learning for Ramp Metering on Highways

Supervised by:

FARHI Nadir

BOUALI Meriem

Realized by:

- CHEHHAT Takieddine
- BENLALA Raid Athmane
- TOUKALI Fares
- KHALFI Ayoub

2024-2025

Table of Contents

1. Introduction

1.1 Background and Motivation

Traffic congestion remains one of the most critical challenges in modern transportation systems, resulting in significant economic, environmental, and social costs. In urban and peri-urban areas, highways are vital arteries for vehicular traffic flow, yet they are often plagued by bottlenecks, particularly at entrance ramps. Ramp metering—a traffic management strategy that regulates the entry of vehicles onto highways using traffic signals—is a proven method to mitigate congestion. Effective ramp metering ensures that traffic entering the highway does not disrupt the mainline flow, thereby improving overall efficiency and reducing delays.

Traditional ramp metering strategies, such as pre-timed or feedback-based approaches, are often inflexible and fail to adapt effectively to dynamic traffic conditions. The advent of artificial intelligence, specifically reinforcement learning (RL), offers a promising alternative by enabling adaptive and data-driven control mechanisms. Reinforcement learning algorithms, such as Q-learning and its extension Deep Q-learning (DQN), have demonstrated significant potential in dynamic decision-making scenarios, including traffic signal control and congestion management. These algorithms allow agents to learn optimal policies through interactions with the environment, leveraging real-time data to minimize delays and improve traffic flow.

1.2 Objective of the Project

The primary objective of this project is to develop and evaluate reinforcement learning-based algorithms for ramp metering control on highways. Specifically, the project aims to:

1. Implement and compare Q-learning and Deep Q-learning (DQN) algorithms for traffic light control at a highway ramp.
2. Optimize traffic flow on both the highway mainline and the entry ramp, minimizing congestion and delays.
3. Evaluate the effectiveness of RL-based ramp metering in scenarios with and without traffic lights, using the SUMO (Simulation of Urban Mobility) traffic simulator.

This project serves as a bridge between theoretical advancements in RL and their practical applications in intelligent transportation systems, contributing to the development of adaptive traffic management solutions.

2.3 Scope and Challenges

The scope of this project involves creating a simulation environment that models a highway segment with an entry ramp controlled by traffic signals. The RL agent will operate within this environment, observing traffic conditions, taking actions to control the ramp traffic light, and receiving rewards based on the efficiency of the resulting traffic flow. Key challenges to address include:

- **State Representation:** Defining a comprehensive state space that captures critical traffic parameters on both the highway and ramp.
- **Reward Design:** Formulating a reward function that balances competing objectives, such as minimizing ramp waiting times while maintaining optimal highway flow.
- **Algorithm Convergence:** Ensuring that the RL algorithms converge to effective policies, particularly in the case of DQN, which involves training a neural network.
- **Simulation Complexity:** Managing the computational demands of running realistic traffic simulations in SUMO, including the calibration of vehicle behaviors and traffic demand.

2.4 Significance and Contributions

The integration of reinforcement learning with ramp metering represents a significant step toward intelligent and adaptive traffic control systems. This project contributes to the field by:

1. Demonstrating the application of Q-learning and DQN in a ramp metering context.
2. Providing insights into the design of RL-based traffic control systems, including state representation, reward formulation, and action space definition.
3. Evaluating the performance of RL-based models against baseline scenarios, offering practical recommendations for deployment.

The findings from this project have the potential to inform real-world implementations of adaptive traffic control, paving the way for smarter and more efficient transportation networks.

2. Brief Explanation of Q-Learning and DQN Algorithms

2.1 Q-Learning:

Q-learning is a model-free reinforcement learning algorithm used to find the optimal action-selection policy for a given finite Markov Decision Process (MDP). The algorithm is based on the idea of learning the Q-value, or quality, of state-action pairs, which represents the expected utility of taking a specific action in a given state and following the optimal policy thereafter. The Q-value is updated iteratively using the Bellman equation:

Where:

- : Current Q-value for state and action .
- : Learning rate.
- : Reward received after taking action in state .
- : Discount factor for future rewards.
- : Next state resulting from action .

Q-learning does not require a model of the environment and can handle environments with stochastic transitions and rewards. It is particularly effective in scenarios with discrete state and action spaces.

Pseudocode for Q-Learning Algorithm:

```
Initialize Q-table with random values
For each episode:
    Initialize state s
    While s is not terminal:
        Choose action a using an epsilon-greedy policy
        Take action a, observe reward r and next state s'
        Update Q-value:
            Q(s, a) = Q(s, a) + alpha * (r + gamma * max_a' Q(s', a') - Q(s, a))
        Update state s = s'
End
```

2.2 Deep Q-Learning (DQN):

Deep Q-Learning extends Q-learning by using a deep neural network to approximate the Q-value function, enabling the algorithm to handle high-dimensional and continuous state spaces. Instead of maintaining a Q-table, the DQN algorithm uses a neural network to predict Q-values for all possible actions given a state. Key innovations in DQN include:

1. **Experience Replay:** Stores transitions (state, action, reward, next state) in a replay buffer and samples them randomly during training to break the correlation between consecutive experiences, improving stability and convergence.
2. **Target Network:** Maintains a separate target network to provide stable target Q-values during training, reducing oscillations.
3. **Loss Function:** Uses a mean squared error loss between the predicted Q-values and target Q-values:

Where represents the parameters of the current network, and represents the parameters of the target network.

DQN has been successfully applied to various domains, such as gaming, robotics, and traffic control, due to its ability to learn directly from raw state representations like images or complex numerical data.

<u>**Pseudocode for Deep Q-Learning (DQN) Algorithm:**</u>

```
Initialize replay buffer D
Initialize Q-network with random weights theta
Initialize target Q-network with weights theta- = theta
For each episode:
    Initialize state s
    While s is not terminal:
        Choose action a using epsilon-greedy policy with Q-network
        Take action a, observe reward r and next state s'
        Store transition (s, a, r, s') in replay buffer D
        Sample a random minibatch from D
        For each transition (s, a, r, s') in the minibatch:
            Compute target:
                y = r + gamma * max_a' Q(s', a'; theta-)
            Compute loss:
                L = (y - Q(s, a; theta))^2
        Perform gradient descent on L to update theta
        Update target network weights: theta- = theta
        Update state s = s'
End
```

2.3 Summarized Differences Between Q-Learning, Deep Q-Learning, and Deep Q-Network:

The following figure illustrates the differences between Q-Learning and Deep Q-Learning:

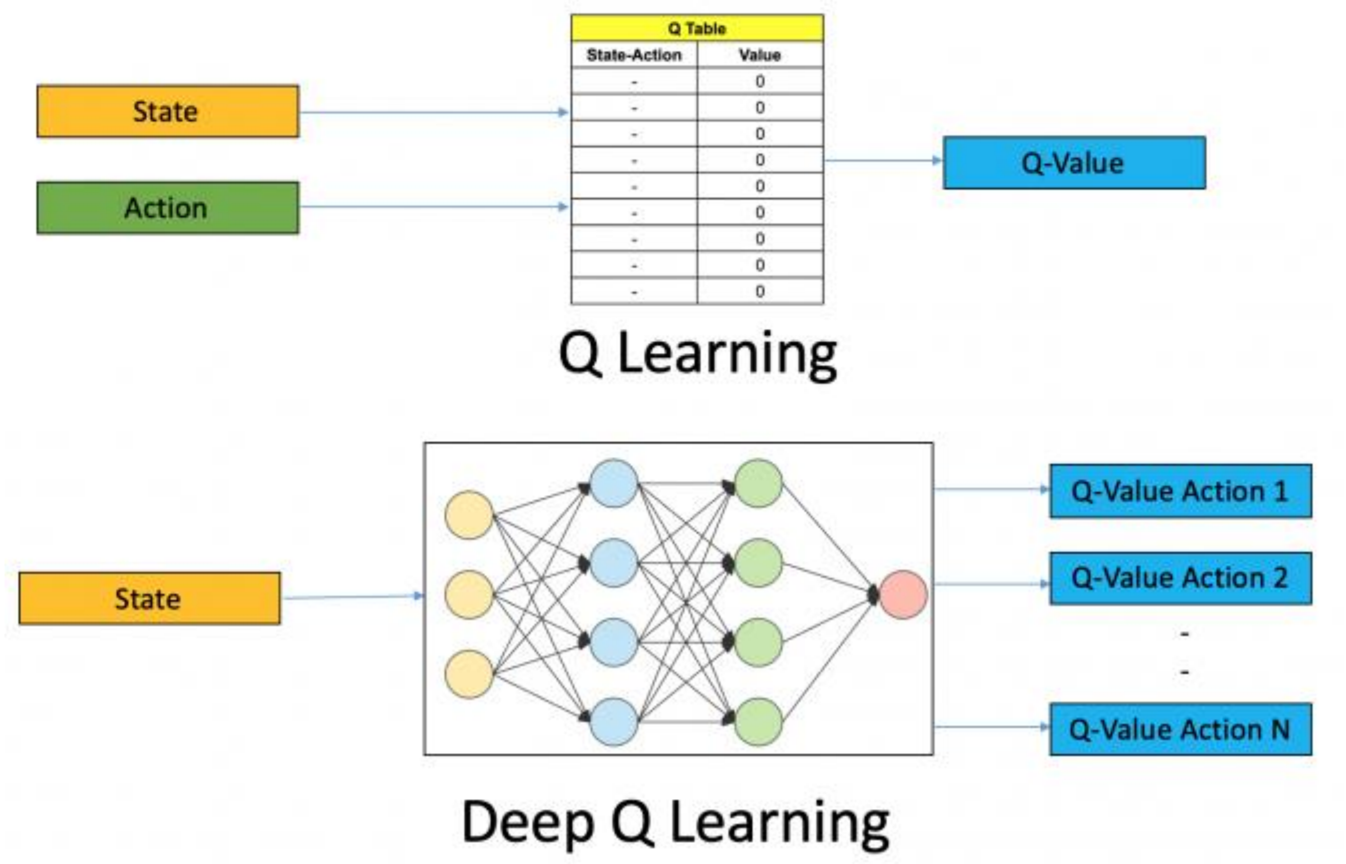


Figure 1 : Difference between Q-Learning and Deep-Learning

3. Description of Simulations

Here are the details about the Simulation Environment:

1. **Number of Lanes:**
   - The highway segment consists of 3 mainline lanes.
   - The entry ramp has a single lane controlled by a traffic light.
2. **Car-Following Model:**
   - The simulation uses the Intelligent Driver Model (IDM) to simulate the car-following behavior of vehicles. IDM accounts for safe distances, speed adjustments, and acceleration/deceleration dynamics.
3. **Lane-Changing Modes:**
   - Vehicles follow a strategic and mandatory lane-changing model, ensuring smooth merges from the ramp to the mainline.
   - Discretionary lane changes are allowed to optimize traffic flow and avoid congestion.
4. **Types of Vehicles:**
   - The simulation includes a mix of passenger cars and heavy vehicles (e.g., trucks), with passenger cars making up 80% of the traffic.
   - Vehicle parameters such as maximum speed, acceleration, and length are defined based on realistic distributions.
5. **Road Length:**
   - The highway segment is 1 km long.
   - The entry ramp is 200 meters long.
6. **Traffic Demand:**
   - Freeway Traffic: The mainline traffic demand ranges from 800 to 2000 vehicles per hour.
   - Ramp Traffic: The ramp demand ranges from 100 to 500 vehicles per hour.
   - Traffic arrivals follow a Poisson process to simulate realistic traffic patterns.

These parameters provide a controlled and realistic environment to evaluate the performance of Q-learning and DQN-based ramp metering strategies.

4. Detailed RL Problem Reformulation

State Definition: The state representation includes critical traffic parameters to provide the RL agent with a comprehensive view of the system:

1. **Highway State:**
   - Number of vehicles in each lane of the highway.
   - Average speed of vehicles in each lane.
   - Density of vehicles over a defined segment of the highway.
2. **Ramp State:**
   - Number of vehicles waiting on the ramp.
   - Average waiting time for vehicles on the ramp.
   - Current traffic light phase (green or red).

This information enables the agent to understand both local and global traffic conditions, facilitating informed decision-making.

Reward Function: The reward function is designed to optimize traffic flow on both the highway and the ramp. Key objectives include:

1. **Minimizing Ramp Waiting Time:**
   - Negative reward proportional to the average waiting time of vehicles on the ramp.
2. **Maintaining Highway Flow:**
   - Negative reward for high vehicle density or low average speed on the highway.
3. **Encouraging Efficient Phase Changes:**
   - Small negative reward for frequent phase changes to avoid unnecessary delays due to switching.

The overall reward at each time step is computed as: Where:

- : Average waiting time on the ramp.
- : Vehicle density on the highway.
- : Average speed of vehicles on the highway.
- : Tunable weights to balance the objectives.

Action Space: The action space defines the possible decisions the agent can make:

1) **Traffic Light Control:**
   a) Switch the traffic light to green (allow ramp vehicles to enter the highway).
   b) Switch the traffic light to red (stop ramp vehicles from entering the highway).

2) **Phase Duration Adjustment:**
   a) Extend the current green or red phase.

These discrete actions enable the agent to dynamically control the ramp metering process, adapting to changing traffic conditions.

5. Hyperparameters of Each Algorithm and ANN Architecture in the Case of DQN

5.1 Q-Learning Hyperparameters:

- Learning Rate: 0.1
- Discount Factor: 0.9
- Exploration Rate: Decays from 1.0 to 0.1 over 1000 episodes.
- Number of Episodes: 2000
- Maximum Steps per Episode: 200

5.2 DQN Hyperparameters:

- Neural Network Architecture:
  - Input Layer: Size matches the state representation (e.g., 5-10 neurons depending on traffic features).
  - Hidden Layers: Two fully connected layers with 64 neurons each, ReLU activation.
  - Output Layer: Size matches the action space (e.g., 2 for green/red traffic light control).
- Learning Rate: 0.0001 (Adam optimizer).
- Discount Factor: 0.95
- Replay Buffer Size: 100,000 transitions
- Batch Size: 32
- Target Network Update Frequency: Every 1000 steps
- Exploration Rate: Decays from 1.0 to 0.1 over 5000 steps.

6. Brief Description of the Code

1. **Required Libraries:**
   - Python libraries: NumPy, TensorFlow/PyTorch (for DQN), Matplotlib, SUMO-Traci.
2. **Project Structure:**

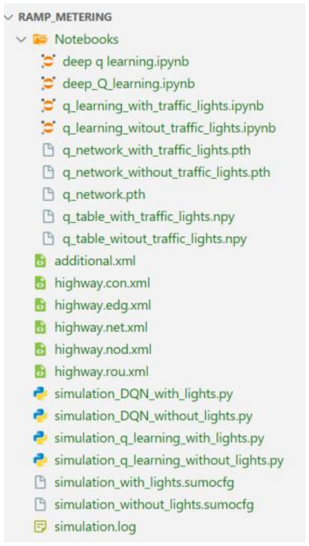
Notebooks

deep_q_learning.ipynb: contains the implementation and experimentation of the Deep Q-Learning (DQN) algorithm.

deep_Q_learning.ipynb: Another notebook for DQN.

q_learning_with_traffic_lights.ipynb: Implements Q-Learning for the case where traffic lights are included on the ramp.

q_learning_without_traffic_lights.ipynb: Implements Q-Learning for the scenario without traffic lights on the ramp.



```
RAMP_METERING
  Notebooks
    deep q learning.ipynb
    deep_Q_learning.ipynb
    q_learning_with_traffic_lights.ipynb
    q_learning_witout_traffic_lights.ipynb
    q_network_with_traffic_lights.pth
    q_network_without_traffic_lights.pth
    q_network.pth
    q_table_with_traffic_lights.npy
    q_table_witout_traffic_lights.npy
  additional.xml
  highway.con.xml
  highway.edg.xml
  highway.net.xml
  highway.nod.xml
  highway.rou.xml
  simulation_DQN_with_lights.py
  simulation_DQN_without_lights.py
  simulation_q_learning_with_lights.py
  simulation_q_learning_without_lights.py
  simulation_with_lights.sumocfg
  simulation_without_lights.sumocfg
  simulation.log
```

Model Files

q_network_with_traffic_lights.pth: Saved neural network model for the DQN algorithm with traffic lights.

q_network_without_traffic_lights.pth: Saved neural network model for the DQN algorithm without traffic lights.

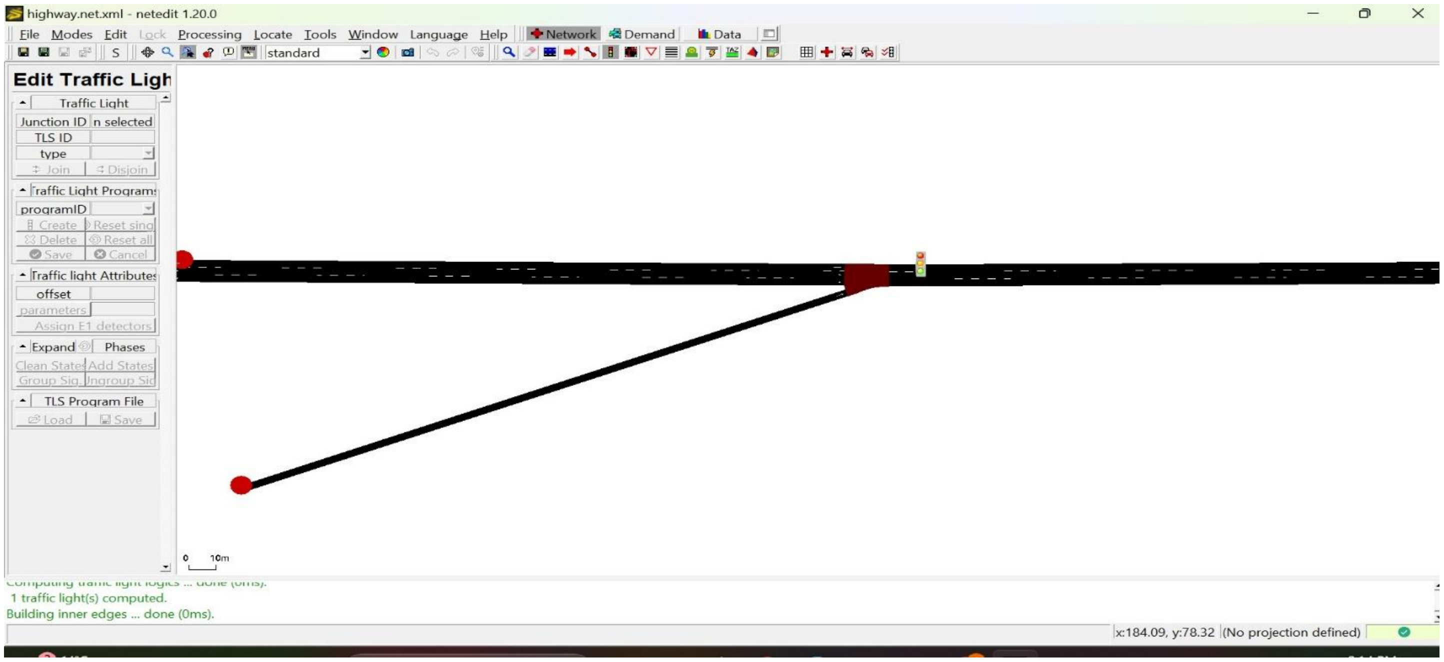
q_network.pth: Generic saved model, likely for experimentation.

q_table_with_traffic_lights.npy: The Q-table generated for the case with traffic lights using traditional Q-Learning.

q_table_without_traffic_lights.npy: The Q-table for the case without traffic lights using Q-Learning.

SUMO Simulation Files

XML Files: (used for traffic simulation configuration)

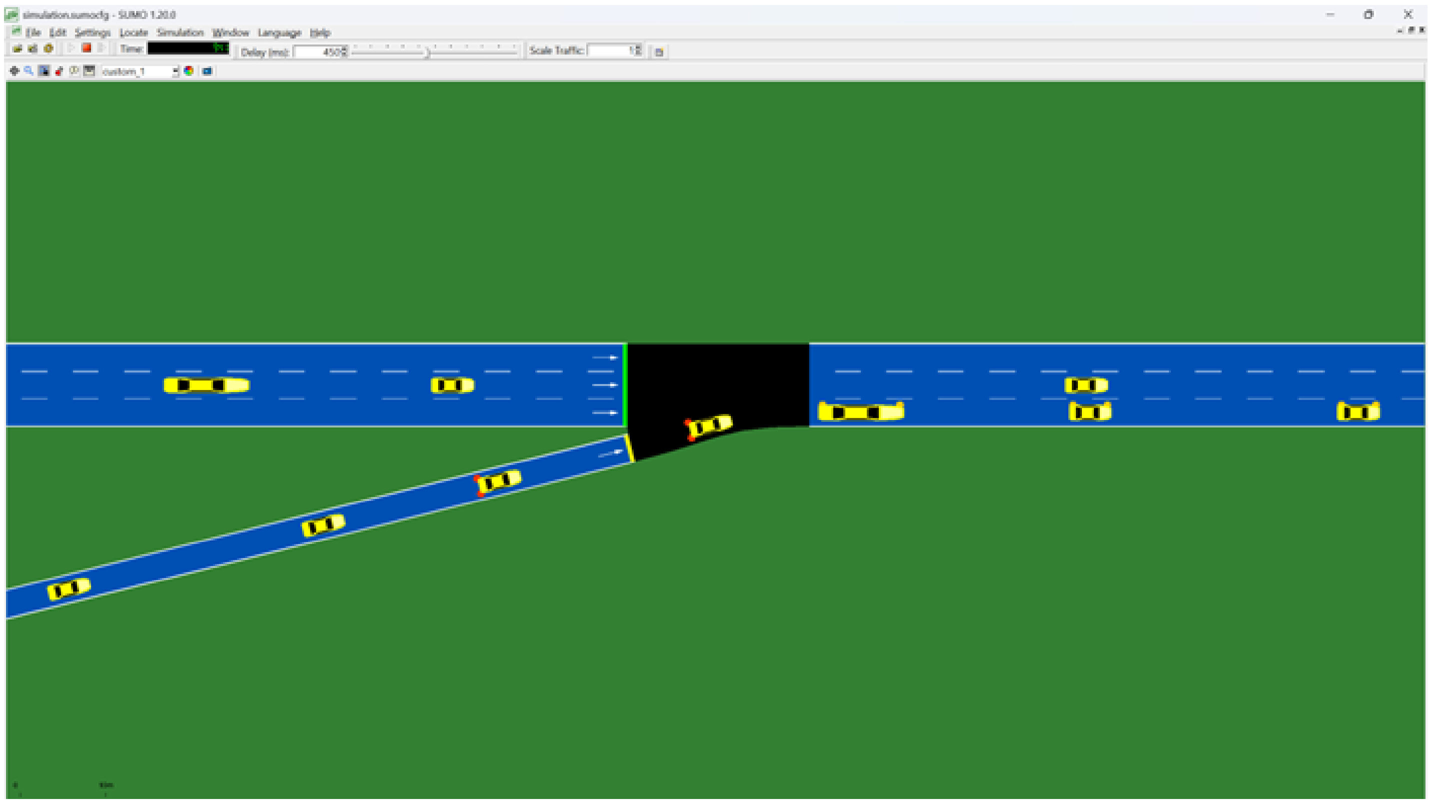


```
Computing traffic light logics ... done (0ms).
1 traffic light(s) computed.
Building inner edges ... done (0ms).
```

x:184.09, y:78.32 (No projection defined)

additional.xml: Defines additional elements for the SUMO simulation (e.g., routes, detectors, traffic lights).
highway.con.xml: Connectivity details for the network (connecting edges and nodes).
highway.edg.xml: Defines the edges (road segments).
highway.net.xml: The compiled SUMO network file.
highway.nod.xml: Defines the nodes (intersections or points of interest).
highway.rou.xml: Specifies vehicle routes and traffic flows.

SUMO Configuration Files


simulation.sumocfg - SUMO 1.20.0
File Edit Settings Locate Simulation Window Language Help
Time:
Delay (ms): 450
Scale Traffic: 1
custom_1

simulation_with_lights.sumocfg: Configuration for SUMO with traffic lights.
simulation_without_lights.sumocfg: Configuration for SUMO without traffic lights

Simulation Scripts

simulation_DQN_with_lights.py: Executes the DQN-based simulation with traffic lights.
simulation_DQN_without_lights.py: Executes the DQN-based simulation without traffic lights.
simulation_q_learning_with_lights.py: Executes Q-Learning simulation with traffic lights.
simulation_q_learning_without_lights.py: Executes Q-Learning simulation without traffic lights.

Log Files

simulation.log: Likely stores logs or output from simulation runs.

3. Execution Instructions

- **Configure SUMO Simulation**:
    - Modify simulation parameters in the respective SUMO configuration files:
        - simulation_with_lights.sumocfg: For simulations with traffic lights.
        - simulation_without_lights.sumocfg: For simulations without traffic lights.
- **Run Q-Learning**:
    - **With Traffic Lights**:

```
python simulation_q_learning_with_lights.py
```

  - **Without Traffic Lights**:

```
python simulation_q_learning_without_lights.py
```

- **Run Deep Q-Learning (DQN)**:
  - **With Traffic Lights**:

```
python simulation_DQN_with_lights.py
```

  - **Without Traffic Lights**:

```
python simulation_DQN_without_lights.py
```

- **Outputs**:
  - **Saved Models**:
    - q_network_with_traffic_lights.pth: DQN model with traffic lights.
    - q_network_without_traffic_lights.pth: DQN model without traffic lights.
    - q_table_with_traffic_lights.npy: Q-Table for traffic lights.
    - q_table_without_traffic_lights.npy: Q-Table for no traffic lights.
  - **Simulation Logs**: simulation.log
  - **Performance Metrics and Plots**:
    - Look for plots of training rewards or other evaluation metrics saved by the scripts or generated in the notebooks.
- **Visualize Simulation** (Optional):
  - To visualize the simulation in SUMO GUI:
    - Open sumo-gui.
    - Load the appropriate configuration file:
      - simulation_with_lights.sumocfg or simulation_without_lights.sumocfg.
    - Click **Play** to observe the simulation.

7. Evaluation and results: interpretation of different plots

7.1 Q-Learning plots:

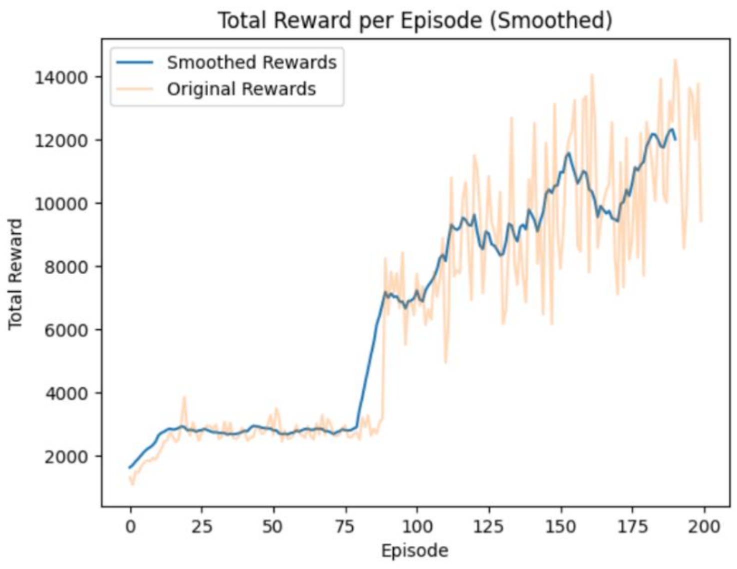


Figure 2- Curve of accumulated reward by episode with

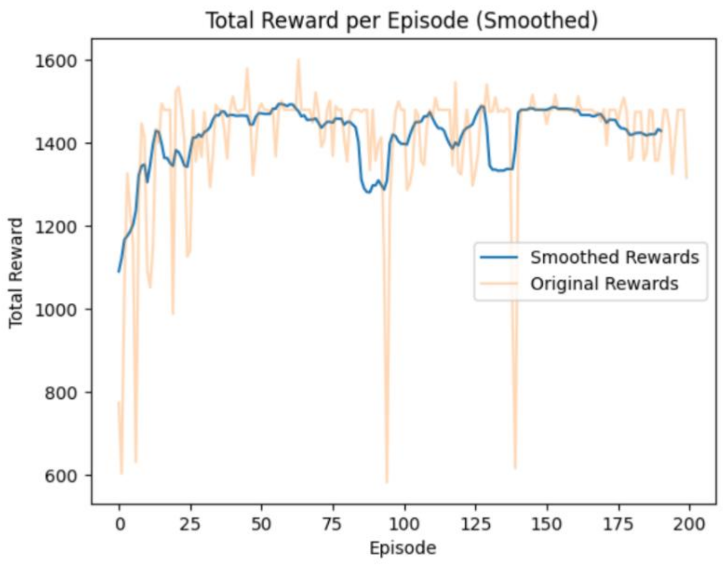


Figure 1- Curve of accumulated reward by episode

7.1.1 Comparative analysis of results:

- **Initial performance**
  - The model without traffic lights starts with lower rewards (774 to 1,066) compared to the model with traffic lights (e.g., 1,310 to 1,487). This indicates that traffic lights initially structure the traffic flow, reducing conflicts
- **Convergence**
  - With traffic lights: Rewards increase significantly after episode 90 (e.g. 8,230, 6,460) and reach high values from episode 156 (13,241).
  - Without traffic lights: Although there is an improvement (e.g. 1 326, 1494), rewards remain significantly lower even in late episodes.
- **Stability**
  - With traffic lights: Rewards become relatively stable and high.
  - Without traffic lights: Rewards are less regular, with more variability.
- **Impact on traffic**
  - With traffic lights: The flow efficiency is significantly higher (15.7 vehicles/hour) and the average waiting time is zero.

    - Without traffic lights: The results are expected to be lower due to the lack of regulation. The flow efficiency is (6.91 vehicles/hour) and the average waiting time is 4.62 seconds.

7.1.2 Key Insights:

- Traffic lights help organize traffic, which improves the cumulative reward and stability of the system.
- Although the model without lights is simpler to implement, it fails to match the performance of the model with lights.
  Recommendations:
- Using traffic lights powered by Q-Learning is best for complex environments with high traffic flows.
- For low density areas, a no-fire approach might be sufficient.

7.2 DQN plots:

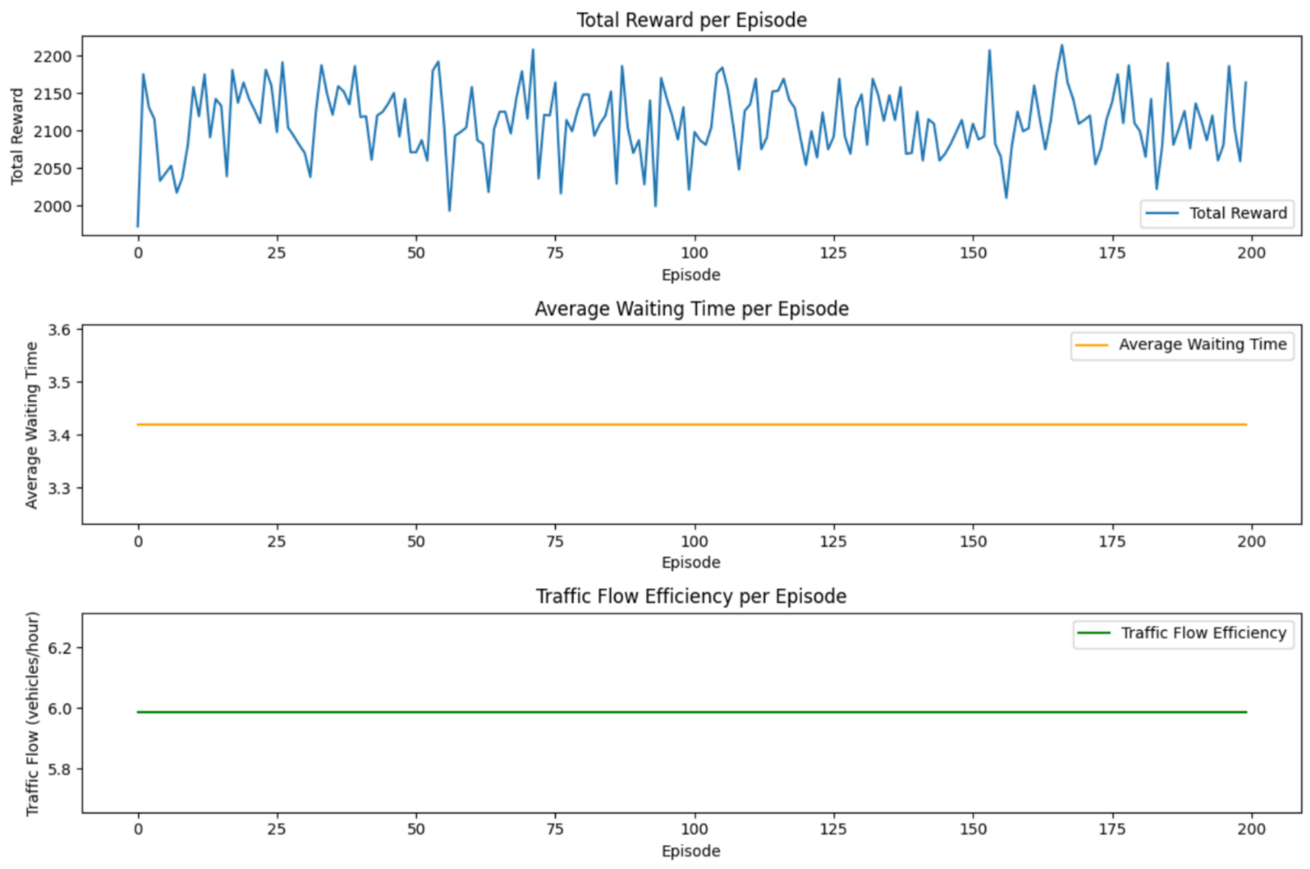


Figure 3- Curves without traffic lights

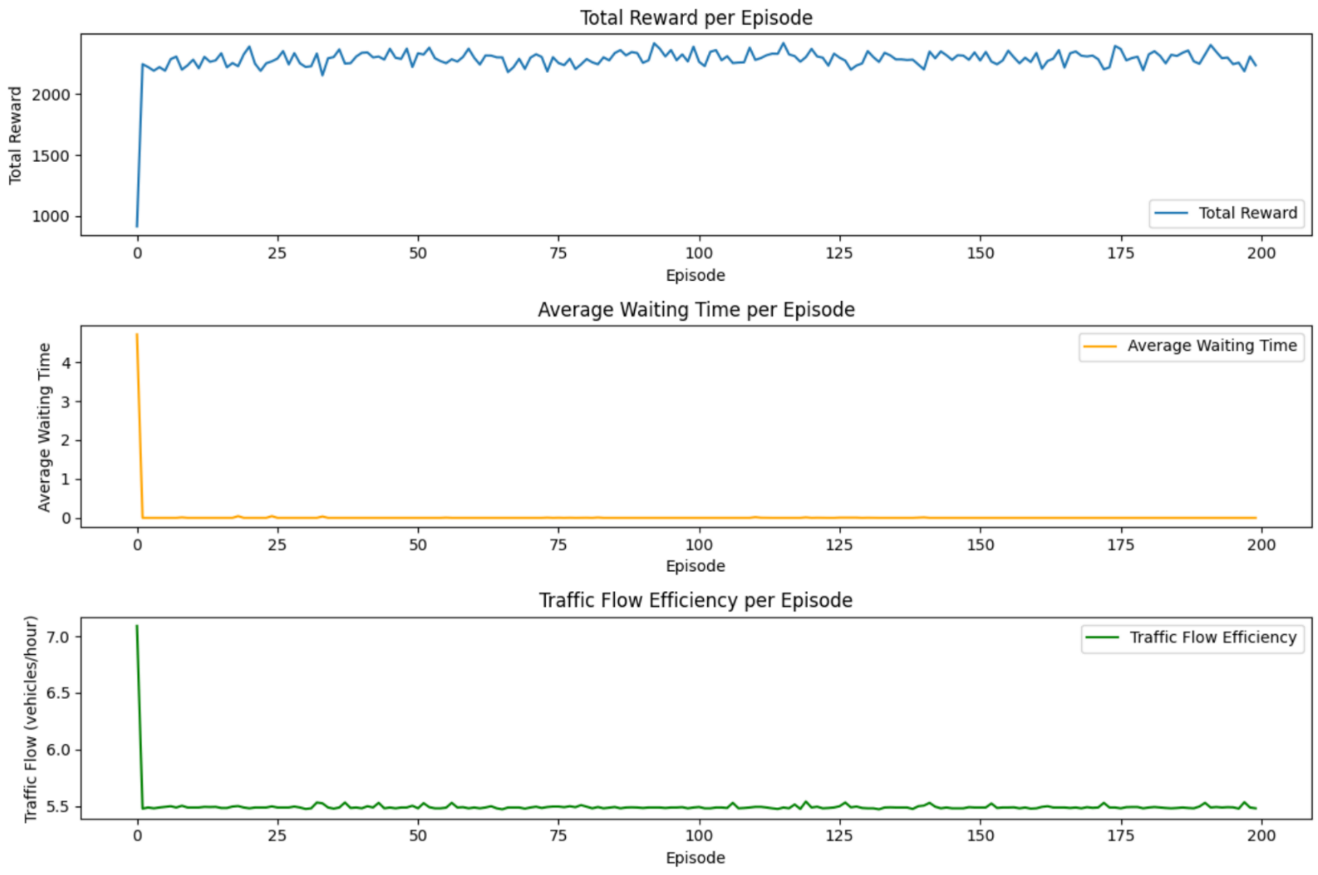


Figure 4- Curves with traffic lights

7.2.1 Comparative analysis of results:

- **Initial performance**
  - With traffic lights: The initial reward is high, indicating good initial traffic control.
  - Without traffic lights: The initial reward is slightly lower, suggesting initial difficulties in handling traffic efficiently.
- **Convergence**
  - With traffic lights: The curve shows rapid convergence towards high rewards, with notable stability after the first episodes.
  - Without traffic lights: there is no convergence!

- **Stability**
  - With traffic lights: Rewards are stable at a high level after convergence.
  - Without traffic lights: Stability is observed, but fluctuations are slightly more frequent.
- **Impact on traffic**

    - With traffic lights: Efficient management significantly reduces average waiting time and improves traffic flow efficiency.
    - Without traffic lights: Average waiting time remains higher, and traffic flow efficiency is slightly lower.

7.2.2 Key Insights:

- Traffic lights play a crucial role in optimizing traffic flow. With Deep Q-Learning, they enable dynamic regulation.
- Efficient, reducing waiting time and increasing overall traffic flow, especially in complex environments.

  Recommendations:
- Using traffic lights powered by Deep Q-Learning is best for complex environments with high traffic flows.
- For low density areas, a no-fire approach might be sufficient.

8. Conclusion

This project investigated the application of reinforcement learning techniques, specifically Q-Learning and Deep Q-Networks (DQN), for optimizing ramp metering in highway traffic management. By simulating traffic scenarios with and without traffic lights, we aimed to evaluate the adaptability and efficiency of these algorithms in handling dynamic traffic conditions.

The results revealed that both Q-Learning and DQN can effectively optimize traffic flow, with distinct strengths. Q-Learning, being simpler to implement, provided a practical solution for environments with moderate complexity. On the other hand, DQN exhibited superior adaptability and scalability, effectively managing the higher-dimensional state spaces and dynamic traffic behaviors associated with complex environments.

Traffic lights played a pivotal role in enhancing system performance, as their inclusion enabled better regulation of vehicle entry onto highways. Simulations demonstrated that models with traffic lights significantly outperformed those without, achieving higher cumulative rewards, reduced waiting times, and improved overall traffic flow. These findings highlight the importance of integrating traffic lights with intelligent control systems to capture real-world dynamics and improve system reliability.

Beyond technical outcomes, this project demonstrates the broader potential of reinforcement learning to revolutionize traffic management by offering data-driven, adaptive solutions to congestion. Compared to traditional rule-based methods, reinforcement learning provides a more flexible framework that can adjust to real-time traffic conditions, reducing bottlenecks and improving the overall efficiency of urban transportation systems.

Future work could explore several avenues to build on this foundation. Real-world validation of the proposed models would provide insights into their practical utility and robustness. Expanding the scope to include multi-ramp or multi-intersection systems could offer a more holistic approach to urban traffic management. Additionally, incorporating advanced reward functions and multi-agent learning could further optimize decision-making processes and accommodate the complexity of modern traffic systems.

By bridging theoretical advancements in reinforcement learning with practical transportation challenges, this project contributes to the development of smarter, more adaptive traffic control systems that have the potential to transform the way we manage road networks.