Reinforcement Learning and Optimal Control

Completed by :

Dhia Aderssa
Omar M'rad
Maissa Bouzid
Riadh Rabti

Subject :

*** Reinforcement learning for ramp metring on highways ***

Academic Year :

2023 - 2024

ABSTRACT :

Amongst a growing need for a solution to help mitigate the downsides of traffic congestion and its economic , social and environmental tolls ,our project aims to leverage the potential of Artificial intelligence in general and Reinforcement learning in particular in automating traffic light control in a more efficient way .
The project focuses on applying reinforcement learning techniques such as Deep Q learning on a traffic light linking a ramp and a highway.
We used a Simulation environment using SUMO software in order to best calibrate and test different hypothesis .
We set specific simulation environment parameters and noted the results from a classic traffic light manual lights cycle and used it as a reference simulation ,Furthermore we tried the reinforcement learning technique to compare the different results and draw conclusions and findings .
We were able to apply the algorithm successfully on the traffic light thus having to not only to take actions based on real time data that he receives but to also keep a log of the previous simulations data and improve incrementally .
However , we could not recreate the same conditions of simulation due to software constraints therefore a comparison of the results would be biased .
The initial results of the Simulation using DQN are very promising and show signs of improvement compared to the reference simulation . we aim that through new software updates or new software choice to overcome this minor difficulty.

ACKNOWLEDGMENTS :

We would like to express our deepest gratitude to Dr.Farhi Nadir for his invaluable guidance and profound expertise, which have been pivotal in us achieving our ambitious project on reinforcement learning for ramp metering on highways.
Dr. Nadir's comprehensive understanding and keen insights into this complex field have enlightened us, paving the way for groundbreaking work.
Additionally, our heartfelt appreciation extends to every member of our project team.
Their unique skills, dedication, and collaboration have created an environment that enabled us to achieve our goal and create added value.
Our project stands as an example to what can be accomplished when a group of dedicated and talented individuals unite towards a common goal. We thank each and every one of you for your invaluable contributions and support throughout this 3 month journey, and we hope we have made a meaningful impact in the field of reinforcement learning and traffic management.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter
1 INTRODUCTION

1.1 General context :

The escalating issue of traffic congestion has reached critical proportions globally, posing profound challenges to urban mobility, economic productivity, and environmental sustainability. Traffic congestion has been estimated to cost the global economy hundreds of billions of dollars annually with the US alone losing 179B dollars each year in terms of lost productivity, fuel consumption, and increased transportation costs. On a global scale due to congestion people lose approximately 100 hours a year in traffic and france is even worse as the estimation is 150 hours/year /person .

As cities continue to grow, traditional traffic management systems prove inadequate in dynamically adapting to fluctuating traffic patterns.

1.2 Academic context :

As an academic project for the course Learning and optimal control on the subject of traffic optimization by introducing a reinforcement algorithm . the project is elaborated by a team of 4 students over a period of 3 months .

Chapter 2

SCOPE AND OBJECTIVES

2.1 Problem Statement :

Solutions to help solve or at least mitigate the problem are needed and among them is the use of Reinforcement learning in traffic optimization . In this project we will create a simulation using the software Sumo that mimics a highway with a ramp leading to it . Furthermore , we will use Reinforcement learning algorithms specifically Q-learning and Deep Q-learning to optimize the cycle of the traffic light between the ramp and the highway.

2.2 Objectives :

Through focusing on some evaluation metrics the goal is to help improve traffic with the use of Reinforcement learning algorithm Deep Q-learning . The Goal is to improve and find the best combination in terms of evaluation metrics compared to the initial simulation (reference simulation).

Chapter

3 METHODOLOGY

3.1 Simulation Environment :

3.1.1 SUMO (Simulator of urban Mobility simulator) :

Is an open source, portable, microscopic and continuous multi-modal traffic simulation package designed to handle large networks. We chose as tool to help us with simulating traffic scenarios and urban mobility the software SUMO ,As it has all the features and the

3.1.2 Python and the TRACI Library :

Python, renowned for its versatility and ease of use, is the primary programming language utilized in our project. The integration of the TRACI library within Python enables real-time interaction with SUMO, allowing for the effective implementation of our control strategies.

3.2 Q-learning and Deep Q-learning :

3.2.1 Explain the theoretical foundations of Q-learning and Deep Q-learning :

Machine learning teams employ diverse methods and algorithms to train their models, with Q-learning standing out as a popular choice. Q-learning is a value-based reinforcement learning algorithm utilized to determine the optimal action-selection policy by leveraging a Q function.

In the realm of deep learning, the Q in DQN (Deep Q Network) specifically refers to 'Q-Learning'. DQN is an off-policy temporal difference method, incorporating considerations of future rewards when updating the value function associated with a given State-Action pair. This integration of Q-learning principles in DQN enhances the model's ability to learn and make decisions, making it a widely utilized approach in the field of machine learning.

3.2.2 Q-learning :

Q-learning is a model-free reinforcement learning algorithm employed to determine the optimal action-selection policy within a finite Markov decision process. This algorithm operates by learning the value associated with taking a particular action in a given state. Over time, it aims to maximize these learned values, thereby guiding the decision-making process towards optimal choices for various situations. Through its iterative learning approach, Q-learning equips agents with the ability to adapt and make informed decisions in dynamic environments :

1) Markov Decision Process (MDP) : Q-learning is rooted in the framework of Markov Decision Processes. An MDP models decision-making in situations where an agent interacts with an environment. It consists of states, actions, transition probabilities, and rewards, with the Markov property ensuring that the future state depends only on the current state and action.

2) State-Action Value Function (Q-function) : The Q-learning algorithm revolves around learning a Q-function (state-action value function), denoted as Q(s, a), which represents the expected cumulative reward of taking action "a" in state "s" and following the optimal policy thereafter.

3) Bellman Equation : The core idea is captured by the Bellman equation for Q-values. It expresses the relationship between the current Q-value, immediate reward, and the Q-value of the next state. The equation is recursive, reflecting the sequential decision-making nature of the problem.

3.2.3 Deep Q-learning :

Deep Q-learning (DQN) is an extension of Q-learning that incorporates deep neural networks to handle complex and high-dimensional input spaces. While traditional Q-learning works well for problems with a relatively small state space, it may struggle when dealing with more intricate environments, such as image-based inputs in video games or real-world scenarios.

In Deep Q-learning, a neural network, often referred to as the Q-network, is employed to approximate the Q-function. The Q-network takes the state of the environment as input and outputs Q-values for each possible action. This allows DQN to generalize its learning across a broader range of states.

Key components of Deep Q-learning include :

1) Experience Replay : DQN utilizes experience replay, where past experiences (tuples of state, action, reward, next state) are stored in a replay buffer. During training, random batches of experiences are sampled from this buffer, breaking the temporal correlations in the data and improving the stability of the learning process.

2) Target Q-network : To stabilize the training process further, DQN employs a target Q-network. This is a separate neural network with frozen parameters that is periodically updated with the parameters of the primary Q-network. This helps prevent the learning targets from oscillating and provides a more consistent learning signal.

3) eGreedy Exploration : DQN typically uses an e-greedy strategy for exploration, where with probability e, a random action is chosen, and with probability 1-e, the action with the highest Q-value is selected. This balance between exploration and exploitation is crucial for discovering optimal policies.

3.3 TensorFlow :

In unlocking the potential of Deep Q-learning algorithms, our project strategically employs TensorFlow and Keras as integral components. TensorFlow serves as a scalable and robust foundation for implementing cutting-edge machine learning models, providing the necessary infrastructure for our project's advancements. Meanwhile, Keras serves a crucial role in simplifying the intricate processes involved in constructing and training neural networks. This streamlined approach is indispensable for enhancing the efficiency and effectiveness of our traffic management solutions, allowing us to optimize and fine-tune our system with greater precision.

Chapter

4 IMPLEMENTATION

4.1 System Description and Parameters :

4.1.1 Sumo Building :

4.1.1.1 Edges :

It's a road segment or link in the transportation network. In SUMO, the road network is represented as a collection of edges connecting nodes, where nodes represent intersections or specific points in the transportation system. An edge defines a specific route or path that vehicles can traverse within the simulation. It includes information about the road's characteristics, such as speed limits, lanes, and other attributes that influence vehicle movement.

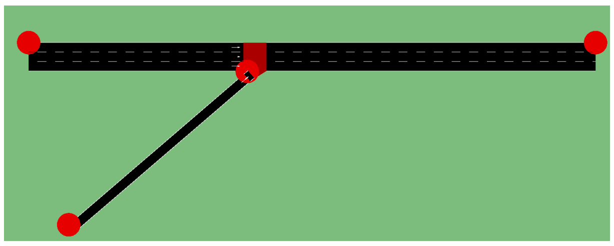


FIGURE 4.1 – Edges

4.1.1.2 Traffic Light :

Traffic light control is strategically implemented at specific junctions, featuring meticulously defined phases with corresponding durations and states. These configurations play a crucial role in orchestrating and optimizing the flow of traffic throughout the network.

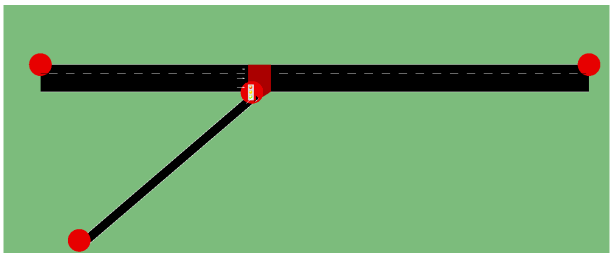


FIGURE 4.2 – Traffic Light

4.1.1.3 Junction and Connection :

In urban planning and transportation systems, the concepts of junctions and connections are instrumental in minimizing traffic congestion and optimizing network efficiency. Junctions, acting as critical nodes, are strategically equipped with traffic control systems to regulate the smooth convergence and divergence of vehicles. Simultaneously, connections, representing the road segments between junctions, are designed to minimize disruptions and ensure seamless transitions. The careful planning and management of junctions and connections are crucial for minimizing congestion, enhancing traffic flow, and fostering a more sustainable and efficient urban environment.

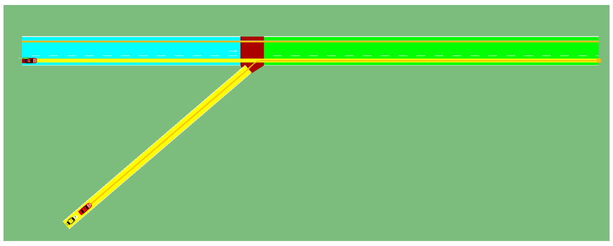


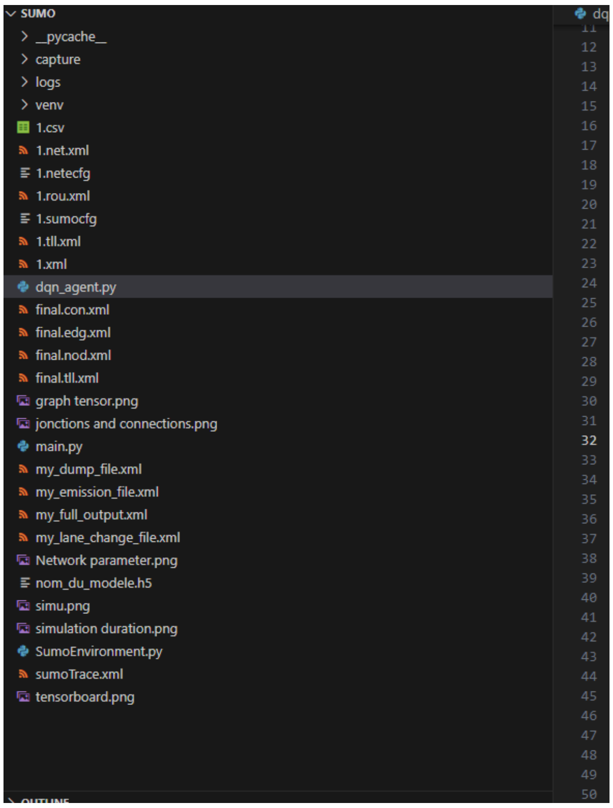
FIGURE 4.3 – Junction and Connection

4.2 Reinforcement Learning implementation :

This code defines a Deep Q-Learning agent (DQNAgent) using the Keras library with TensorFlow backend for reinforcement learning. The agent is designed to learn to play the CartPole-v1 environment from OpenAI Gym.

Let's break down the key components and functionalities of the **DQNAgent** class :

1- Initialization :



```
SUMO
> __pycache__
> capture
> logs
> venv
1.csv
1.net.xml
1.netecfg
1.rou.xml
1.sumocfg
1.tll.xml
1.xml
dqn_agent.py
final.con.xml
final.edg.xml
final.nod.xml
final.tll.xml
graph tensor.png
jonctions and connections.png
main.py
my_dump_file.xml
my_emission_file.xml
my_full_output.xml
my_lane_change_file.xml
Network parameter.png
nom_du_modele.h5
simu.png
simulation duration.png
SumoEnvironment.py
sumoTrace.xml
tensorboard.png
```

FIGURE 4.4 – Project Architecture

The constructor 'init' initializes the agent with parameters such as statesize, actionsize, learningrate, discountfactor,explorationrate, explorationdecay, and memorysize. It creates a neural network model using the private method buildmodel. The agent also maintains a memory buffer (self.memory) using a deque data structure to store experiences for training.

2- Neural Network Model : The private method buildmodel creates a simple feedforward neural network using Keras. It has an input layer with the size of the state space (statesize), two hidden layers with 24 units each and ReLU activation, and an output layer with the size of the action space (actionsize) and linear activation. The model is compiled using mean squared error (mse) loss and the Adam optimizer with a specified learning rate.

3- Experience Replay : The remember method stores experiences (state, action, reward, nextstate, done) in the agent's memory buffer.

4-Action Selection : The chooseaction method selects an action based on the current state using an epsilon-greedy strategy. With probability explorationrate, it chooses a random action; otherwise, it selects the action with the highest predicted Q-value.

5-Replay and Training : The replay method is responsible for training the neural network using randomly sampled experiences from the memory buffer. It uses a target Q-value that incorporates the discount factor and updates the Q-value of the chosen action. The model is trained using the TensorBoard callback for logging training and testing statistics.

6-Training Loop : The train method sets up the CartPole environment and runs the training loop for the specified number of episodes. In each episode, the agent interacts with the environment, stores experiences, and performs experience replay at the end of the episode.

7-Logging and Saving : The agent creates TensorBoard logs for training and testing in separate directories. The model is saved after training using the save method and can be loaded later using the load method.

4.3 Simulation :

4.3.1 Simulation without DQN :

Reference Simulation : The reference simulation is a simulation that we use as reference when comparing results from the test simulations in order to measure the effect of the test simulations . The reference simulation in our case will share many characteristics with the test simulations.

Simulation boundaries : - Simulation duration : 300sc
- Highway parameters : number of lanes : 3 speed limit : 14m/sc initial flow of traffic : 1800 veh/hr highway length : 200m -Ramp parameters : number of lanes : 1 speed limit : 8m/sc initial flow of traffic : 1000 veh/hr Ramp length : 80m -traffic light parameters : Manual with 4 state cycle : -G RRR (42sc) ,Y RRR (3sc) ,R GGG(42sc) , R YYY(3sc)
Simulation Results :
Throughput : 76 veh Delay : 0 sc

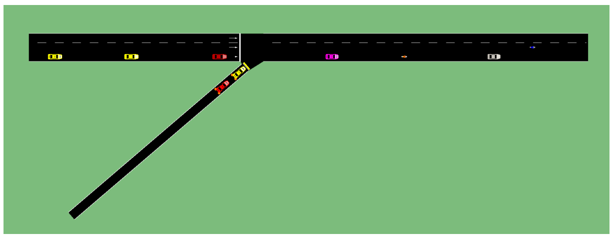


FIGURE 4.5 – Sumo Simulation

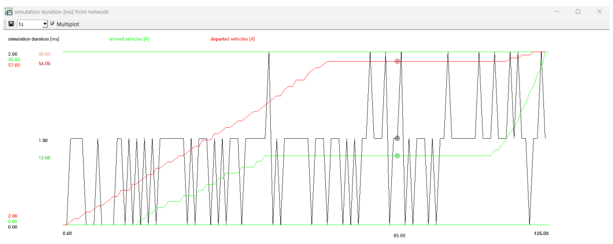
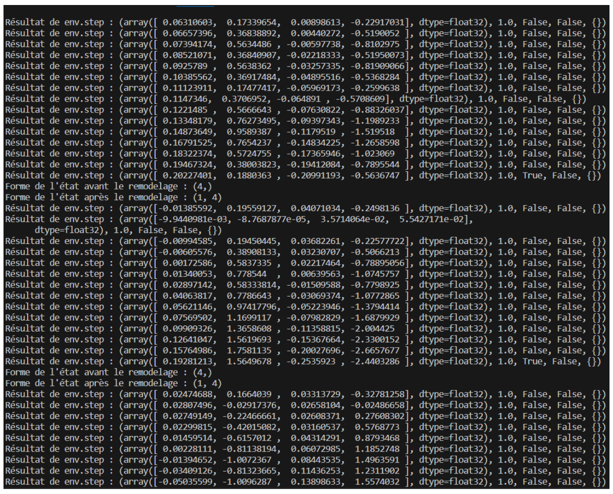


FIGURE 4.6 – Analyze Simulation

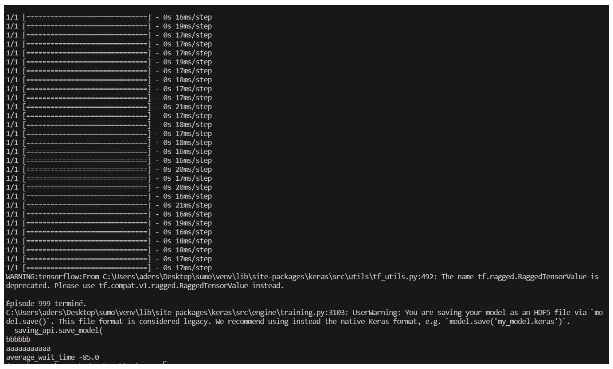
4.3.2 Simulation with DQN :

This graphic showcases the SUMO simulation environment, offering a visual insight into the intersection under the agent's control. This image accentuates the pivotal role of the agent in managing the traffic lights, thereby influencing the overall traffic flow within the simulation. The intersection serves as a focal point where the agent's decisions and strategies directly impact the dynamics of the traffic system, reflecting the significance of intelligent traffic control in SUMO simulations.



```
Résultat de env.step : (array([ 0.06310603,  0.17339654,  0.00898613, -0.22917031], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.06657396,  0.36838892,  0.00440272, -0.5190052 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.07394174,  0.5634486 , -0.00597738, -0.8102975 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.08521071,  0.36840907, -0.02218333, -0.51950073], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.0925789 ,  0.5638362 , -0.03257335, -0.81909066], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.10385562,  0.36917484, -0.04895516, -0.5368284 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.11123911,  0.17477417, -0.05969173, -0.2599638 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.1147346,  0.3706952, -0.064891 , -0.5708609], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.1221485 ,  0.5666643 , -0.07630822, -0.88326037], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.13348179,  0.76273495, -0.09397343, -1.1989233 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.14873649,  0.9589387 , -0.1179519 , -1.519518  ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.16791525,  0.7654237 , -0.14834225, -1.2658598 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.18322374,  0.5724755 , -0.17365946, -1.023069  ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.19467324,  0.38003823, -0.19412084, -0.7895544 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.20227401,  0.1880363 , -0.20991193, -0.5636747 ], dtype=float32), 1.0, True, False, {})
Forme de l'état avant le remodelage : (4,)
Forme de l'état après le remodelage : (1, 4)
Résultat de env.step : (array([-0.01385592,  0.19559127,  0.04071034, -0.2498136 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([-9.9440981e-03, -8.7687877e-05,  3.5714064e-02,  5.5427171e-02],
      dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([-0.00994585,  0.19450445,  0.03682261, -0.22577722], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([-0.00605576,  0.38908133,  0.03230707, -0.5066213 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.00172586,  0.5837335 ,  0.02217464, -0.78895056], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.01340053,  0.778544  ,  0.00639563, -1.0745757 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.02897142,  0.58333814, -0.01509588, -0.7798925 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.04063817,  0.7786643 , -0.03069374, -1.0772865 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.05621146,  0.97417796, -0.05223946, -1.3794414 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.07569502,  1.1699117 , -0.07982829, -1.6879929 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.09909326,  1.3658608 , -0.11358815, -2.004425  ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.12641047,  1.5619693 , -0.15367664, -2.3300152 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.15764986,  1.7581135 , -0.20027696, -2.6657677 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.19281213,  1.5649678 , -0.2535923 , -2.4403286 ], dtype=float32), 1.0, True, False, {})
Forme de l'état avant le remodelage : (4,)
Forme de l'état après le remodelage : (1, 4)
Résultat de env.step : (array([ 0.02474688,  0.1664039 ,  0.03313729, -0.32781258], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.02807496, -0.02917376,  0.02658104, -0.02486658], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.02749149, -0.22466661,  0.02608371,  0.27608302], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.02299815, -0.42015082,  0.03160537,  0.5768773 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.01459514, -0.6157012 ,  0.04314291,  0.8793468 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([ 0.00228111, -0.81138194,  0.06072985,  1.1852748 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([-0.01394652, -1.0072367 ,  0.08443535,  1.4963591 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([-0.03409126, -0.81323665,  0.11436253,  1.2311902 ], dtype=float32), 1.0, False, False, {})
Résultat de env.step : (array([-0.05035599, -1.0096287 ,  0.13898633,  1.5574032 ], dtype=float32), 1.0, False, False, {})
```

FIGURE 4.7 – Execution 1 of the reinforcement learning



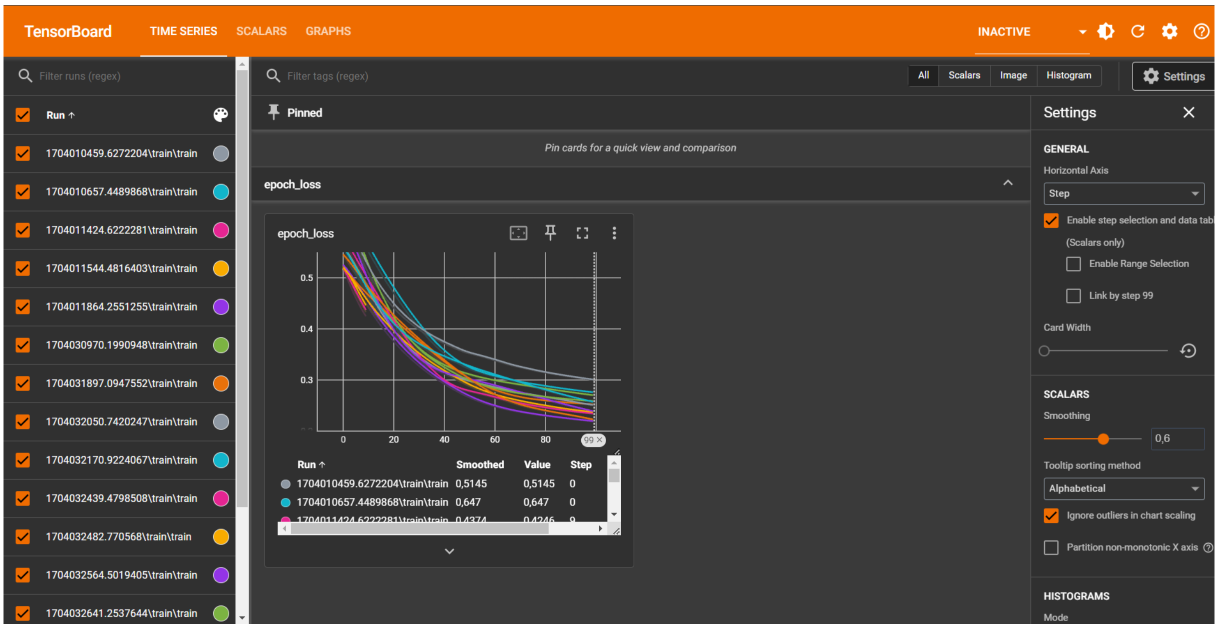
```
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
WARNING:tensorflow:From C:\Users\aders\Desktop\sumo\venv\lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is
deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

Épisode 999 terminé.
C:\Users\aders\Desktop\sumo\venv\lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model as an HDF5 file via `mo
del.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
bbbbbb
aaaaaaaaaaa
average_wait_time -85.0
```

FIGURE 4.8 – Execution 2 of the reinforcement learning

4.3.2.1 TensorBoard :

TensorBoard is a web-based tool provided by TensorFlow for visualizing and analyzing machine learning experiments. It is commonly used to monitor training progress, visualize model graphs, and analyze various metrics such as loss and accuracy. TensorBoard is particularly helpful for gaining insights into the behavior of your machine learning models.

We successfully managed to transform the data from the DQN model into the web tool tensor-board we we were able to visualise the training and the decision made by DQN based on the real time data that it received .
This proved very useful where we noticed the evolution in performance of the DQN model from iteration to the next .



Run ↑
✓ 1704010459.6272204\train\train ●
✓ 1704010657.4489868\train\train ●
✓ 1704011424.6222281\train\train ●
✓ 1704011544.4816403\train\train ●
✓ 1704011864.2551255\train\train ●
✓ 1704030970.1990948\train\train ●
✓ 1704031897.0947552\train\train ●
✓ 1704032050.7420247\train\train ●
✓ 1704032170.9224067\train\train ●
✓ 1704032439.4798508\train\train ●
✓ 1704032482.770568\train\train ●
✓ 1704032564.5019405\train\train ●
✓ 1704032641.2537644\train\train ●

Run ↑	Smoothed	Value	Step
● 1704010459.6272204\train\train	0,5145	0,5145	0
● 1704010657.4489868\train\train	0,647	0,647	0
● 1704011424.6222281\train\train	0,4374	0,4246	9

Settings

GENERAL

Horizontal Axis: Step

✓ Enable step selection and data tab (Scalars only)

☐ Enable Range Selection

☐ Link by step 99

Card Width

SCALARS

Smoothing: 0,6

Tooltip sorting method: Alphabetical

✓ Ignore outliers in chart scaling

☐ Partition non-monotonic X axis

HISTOGRAMS

Mode

FIGURE 4.9 – TensorBoard Graphic

Chapter 5

FINDINGS

We created a reference simulation with static traffic light control and we used throughput and delay as evaluation parameters .
we registered the results of this simulation in order to use it as a baseline for the simulation results of the DQN model .
we could not repeat the same simulation environment conditions as the first simulation due to software constraints .
However we were able to notice the successful implementation of the DQN model in the simulation and the improvements it made from one Simulation to the next .
Furthermore , we noticed better initial results with DQN compared to the reference simulation thanks to the data from the tensor-board .

Chapter
6
CHALLENGED ENCOUNTERED

6.1 Familiarizing with the SUMO Software

6.1.1 Complexity of SUMO :

SUMO (Simulation of Urban Mobility) is a powerful but complex traffic simulation tool. The initial challenge likely involved understanding the diverse features, functionalities, and configurations within SUMO, from defining road networks to simulating traffic dynamics.

6.1.2 Learning Curve :

Given the complexity of SUMO, there may have been a learning curve associated with navigating its interface, understanding its documentation, and efficiently utilizing its capabilities for the specific requirements of your project.

6.2 Finding and Adjusting the DQN Model

6.2.1 Model Selection :

Selecting an appropriate DQN model tailored to your project requirements might have posed a challenge. This involves understanding various DQN architectures, choosing one suitable for traffic management, and considering factors like input representation, network depth, and hyper-parameter tuning.

6.2.2 Adjusting for Traffic Dynamics :

Adapting a generic DQN model for the intricacies of traffic control involves understanding how traffic dynamics impact the learning process. This might include addressing issues related to the scale of state and action spaces, temporal dependencies, and the complexity of the environment.

6.3 Implementing the DQN Model in Traffic Light Configuration Files

6.3.1 Integration with SUMO :

Integrating the DQN model into the SUMO simulation likely presented challenges in terms of aligning the reinforcement learning framework with the simulation environment. This involves modifying or creating traffic light configuration files to accommodate the DQN-based control.

CONCLUSION

In conclusion, the use of reinforcement learning in the context of highway ramp metering control illustrates how artificial intelligence has the ability to significantly improve the way complicated real-world issues are handled. The advantages of utilizing reinforcement learning go beyond traffic optimization ; they provide a preview of a future in which intelligent, adaptive systems will be able to improve many facets of human existence.

USEFUL LINKS

Github Link : https://github.com/Dhia124/Reinforcement-Learning-and-Optimal-Control-Project