Traffic flow Optimization using Reinforcement Learning

Objective

Develop reinforcement learning models to optimize traffic signal timings and comparing the performance of these models with the aim of minimizing average waiting time, maximizing traffic flow and adaptable to dynamic traffic conditions.

Background

We used openstreetmap.org we imported random road map as map.osm file and using the netedit app from the SUMO simulator we generated network.net.xml which is simulator map. And using simulator RandomTrips.py python file we could simulate the traffic system which gives routes.rou.xml and we merged these files and generated the config.sumocfg file got generated. It includes the route configuration and network and verified the setup with the SUMO-gui. Using traci library we established the connection between simulator and gym environment.

Environment

Action space:

For each action there will be change in traffic lights based on the current observation.(green to yellow, yellow to red, red to green)

Observation space:

Each traffic signal represents each agent. Each agent has observation space consists of 3 observation space current signal phase of that junction, no. of vehicles waiting in the junction, average wait time of vehicles waiting in the junction.

Goal:

Our goal is to minimize the average wait time and number of vehicles in each junction.

Terminal state:

Truncation after 1000 timesteps. Once all the cars completed their defined trips before 1000 timesteps then episode get terminated.

Reward:

Reward = (Number of cars waiting * average wait time) * -1 .

As waiting time should be decreased. So its give negative reward.

Modelling

Deep Q Network

Reasons for using Deep Q Network:

• Nonlinear Approximation - DQN uses deep neural networks to approximate the Qfunction, enabling complex state-action mappings.

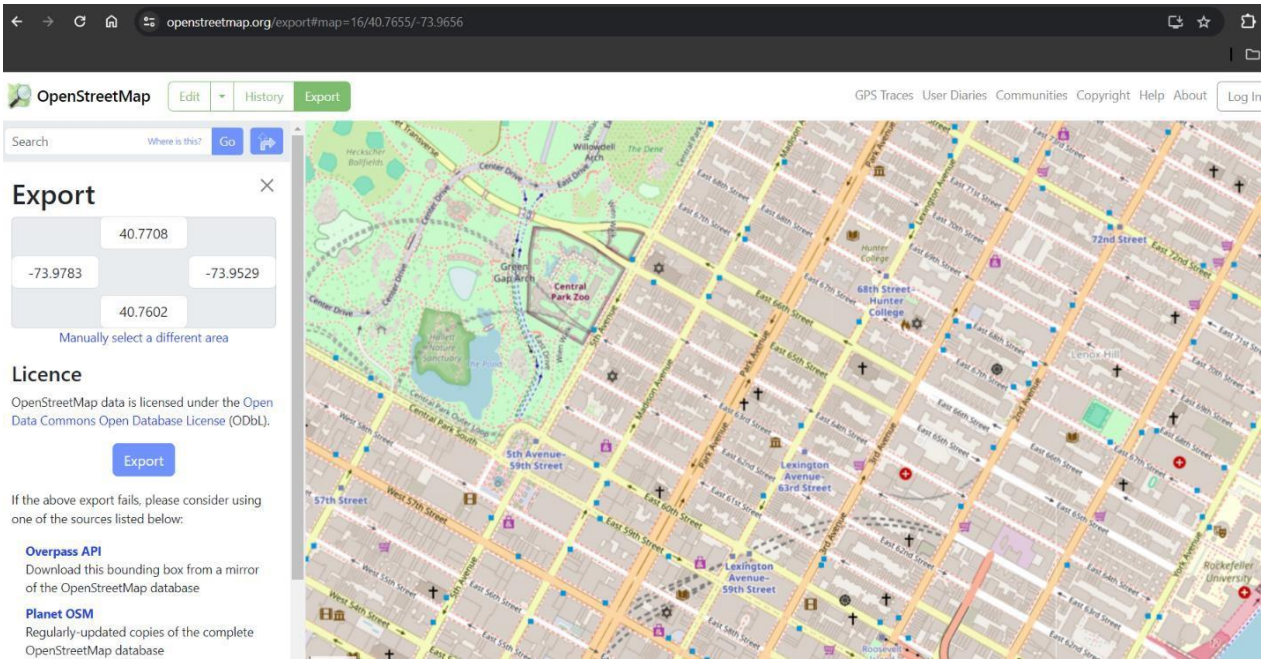
• Experience Replay - Utilizes a replay buffer for efficient learning by storing and randomly sampling past experiences.
• Target Network - Maintains a separate target network to stabilize training and mitigate divergence issues.
• Epsilon-Greedy Exploration - Balances exploration and exploitation by selecting actions greedily with probability ε and randomly with probability (1-ε).

We also tried implementing TD methods but with the given environment there could be possibility of infinite number of state space and hence it makes impossible to implement it for infinite state space environment like this.

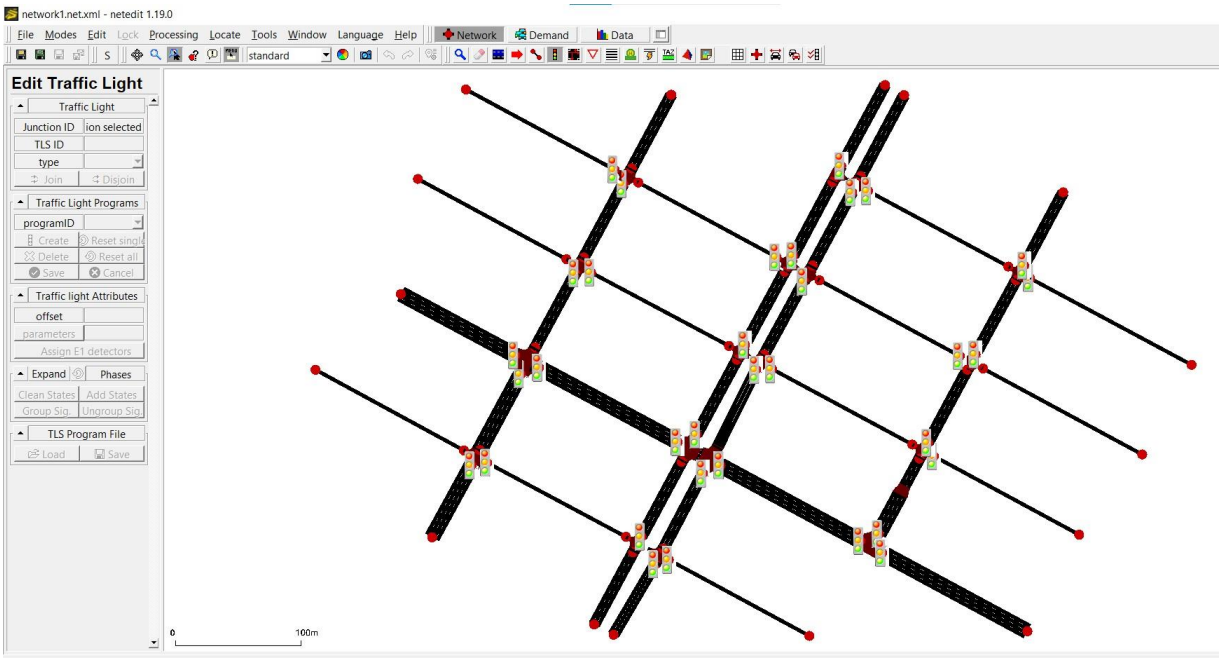
Hyperparameters
replay_start_size: 64
replay_buffer_size: 100000
gamma: 0.99
update_target_frequency: 1000
minibatch_size: 64
learning_rate: 0.001
update_frequency: 1
initial_epsilon: 1.0
final_epsilon: 0.01
max_steps: 1000
loss: Mean Squared Error

Visualization:

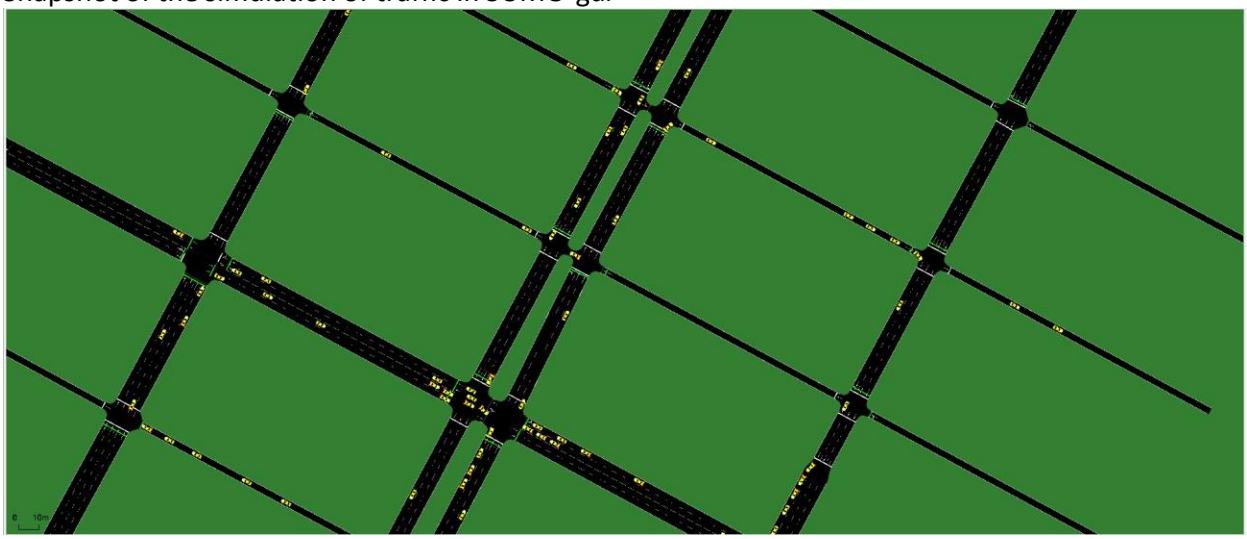
This is the openstreetmap where we exported the real world map for our simulation.



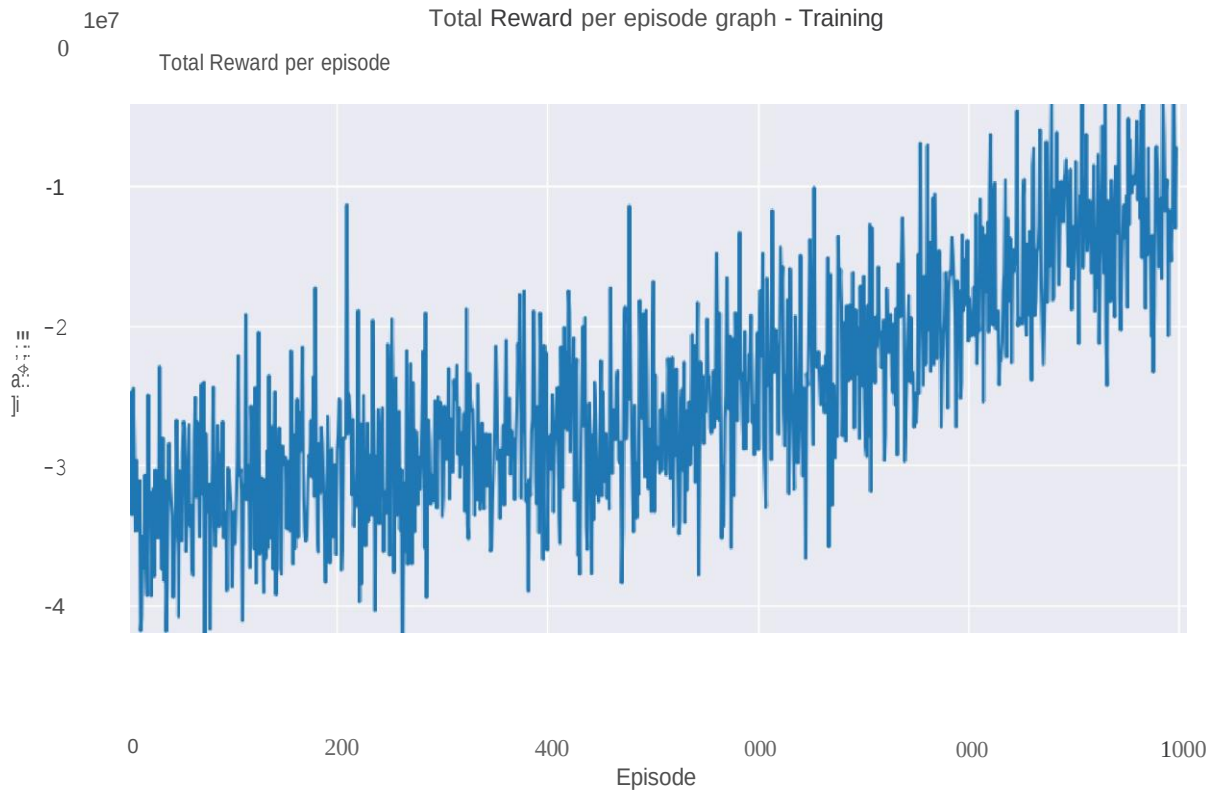
Editing the network.net.xml using netedit app from SUMO.

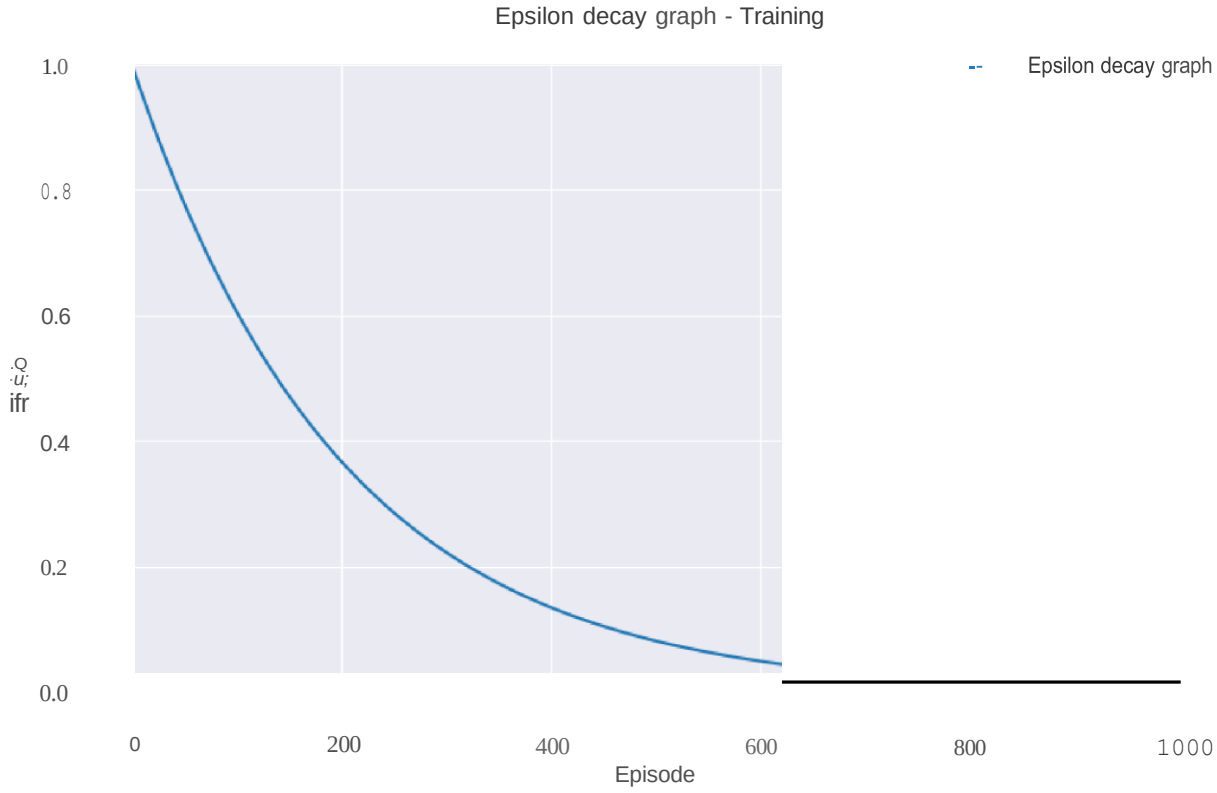


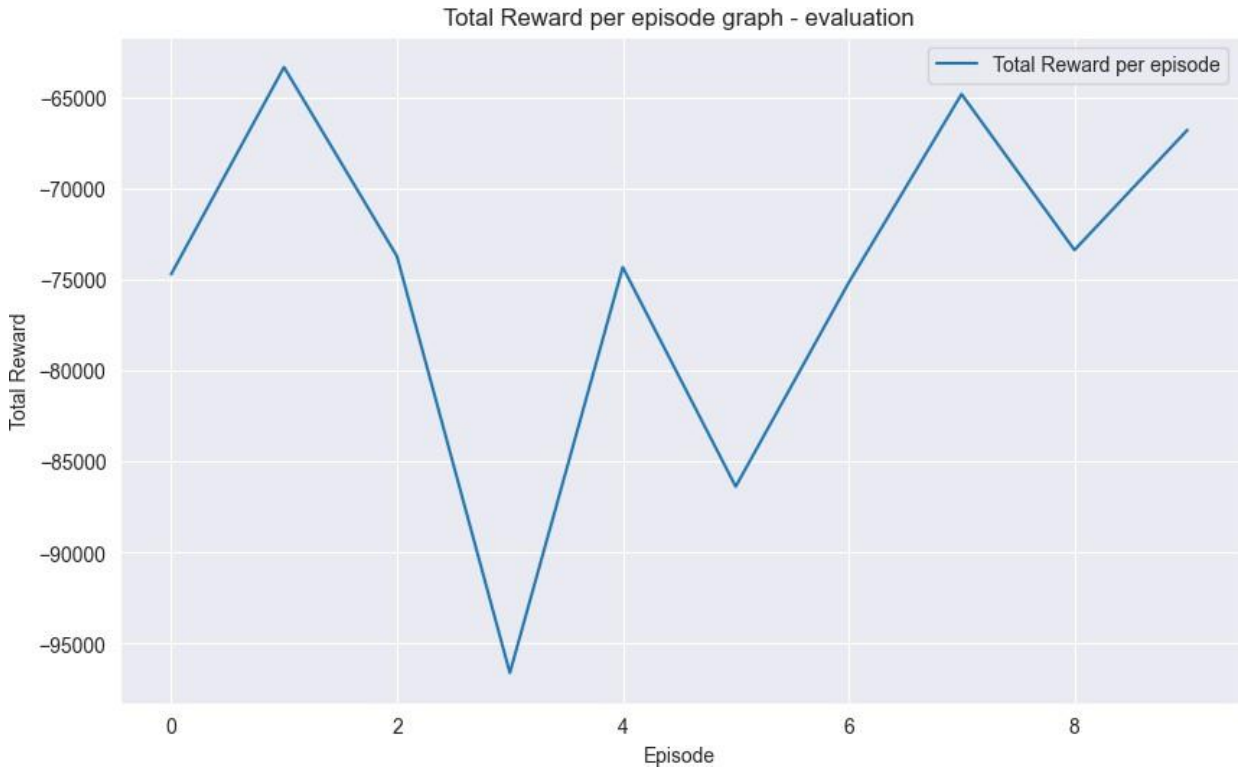
Snapshot of the simulation of traffic in SUMO-gui



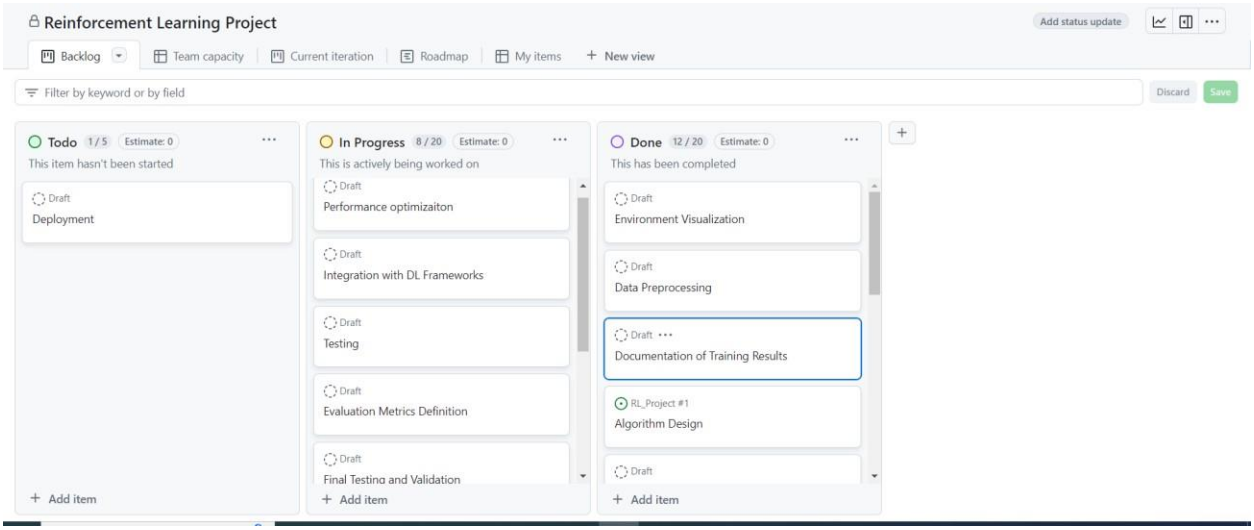
Plots:

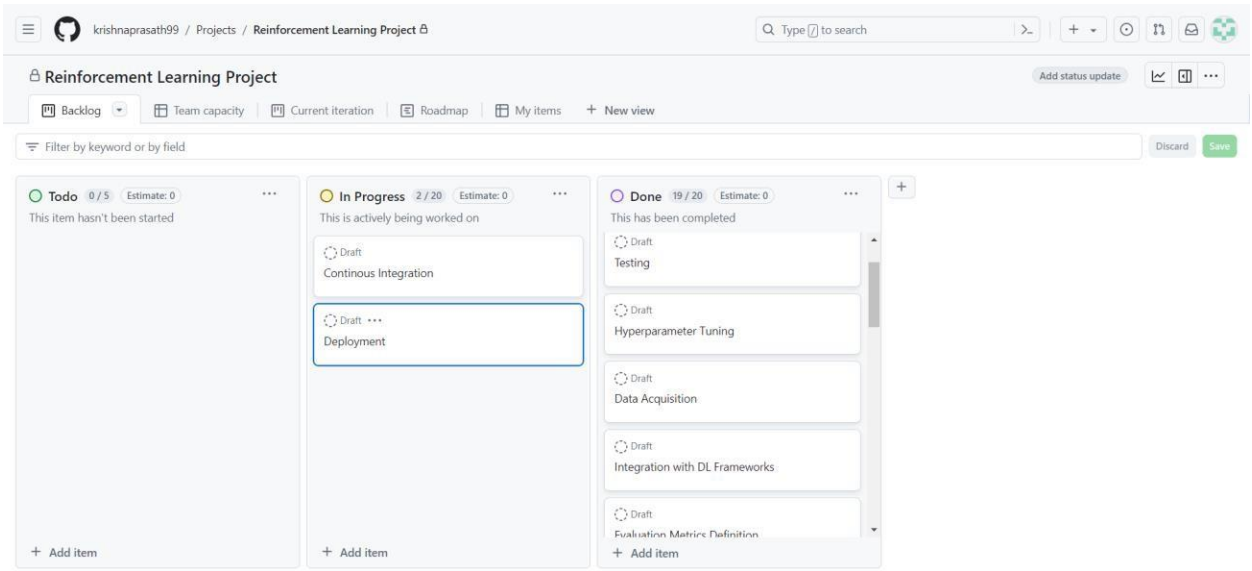

Total Reward per episode graph - Training
1e7
0
-1
-2
-3
-4
Total Reward per episode
0
200
400
000
000
1000
Episode


Epsilon decay graph - Training
Epsilon decay graph
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
Episode



Project management github





Real-World RL Application:

Since we are already implemented the real world Manhattan, New York map, we can implement the same simulation for any part of real world with very little adjustments in the implementation. Real world scenario is to minimize the traffic for any busiest road at any time at any places.

References:

1. T. Chu, J. Wang, L. Codecà and Z. Li, "Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control," in IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 3, pp. 1086-1095, March 2020, doi: 10.1109/TITS.2019.2901791.
2. https://sumo.dlr.de/docs/index.html
3. https://sumo.dlr.de/docs/TraCI.html
4. Also referenced our teammates Assignment 2 DQN model