Reinforcement Learning Project

This project is divided into two parts, (for the Environment Setup part)

Doing a basic custom environment
Mapping the learnings onto a new environment with OpenStreetmap data and integrating into SUMO.

Resources and References:

https://www.openstreetmap.org/

Or use Geofabrik

Timeline

Project Plan Timeline for RL Agent in Traffic Simulation Using SUMO-RL

Target Completion Date: November 25th

Week	Tasks	Expected Time	Comments
Week 1 (Nov 6 - 12)	**Research and Setup**: Explore sumo-rl library and run example environments	3 hours (Mon, Wed, Fri)	Familiarize with sumo-rl and document any initial customization needs.
	Install and configure sumo-rl and SUMO on your system	2 hours (Sat)	Ensure environment is fully functional.
Week 2 (Nov 13 - 19)	**Environment Customization**: Implement initial customization in sumo-rl, focusing on traffic light control logic	2 hours (Mon, Tue)	Limit customization to aspects needed for DQN; additional complexity can be handled in future iterations.
	Set up observation/action spaces for DQN-compatible inputs	2 hours (Thu)	Match traffic light states and vehicle data to DQN agent expectations.
	Early Testing: Test the customized environment with simple agents	3 hours (Sat, Sun)	Confirm environment runs smoothly and logs relevant metrics.
Week 3 (Nov 20	**DQN Implementation & Training**: Integrate DQN agent	3 hours (Mon, Tue)	Start with basic DQN parameters.

Week	Tasks	Expected Time	Comments
- 23)	using PyTorch		
	Implement reward structures to reflect traffic flow and wait times	1 hour (Wed)	Optimize the agent's reward based on traffic efficiency and movement.
Nov 24 - 25	**Final Testing and Adjustments**: Run tests, analyze DQN performance, and fine-tune	3 hours (Fri, Sat)	Ensure the model achieves realistic, stable results before finalizing.

Notes:

- This timeline allocates **1 hour per weekday** and up to **3 hours on weekends** to comfortably finish by **November 25th**.

Updated timelines

Week	Tasks	Expected Time	Comments
Week 1 (Nov 12 - 17)	**Research and Setup**: Explore sumo-rl library and run example environments	3 hours (Mon, Wed, Fri)	Familiarize with sumo-rl and document any initial customization needs.
	Install and configure sumo-rl and SUMO on your system	2 hours (Sat)	Ensure environment is fully functional.
Week 2 (Nov 18 - 24)	**Environment Customization**: Implement initial customization in sumo-rl, focusing on traffic light control logic	2 hours (Mon, Tue)	Limit customization to aspects needed for DQN; additional complexity can be handled in future iterations.
	Set up observation/action spaces for DQN-compatible inputs	2 hours (Thu)	Match traffic light states and vehicle data to DQN agent expectations.
	Early Testing: Test the customized environment with simple agents	3 hours (Sat, Sun)	Confirm environment runs smoothly and logs relevant metrics.
Week 3 (Nov 25 - 30)	**DQN Implementation & Training**: Integrate DQN agent using PyTorch	3 hours (Mon, Tue)	Start with basic DQN parameters.
	Implement reward structures to reflect traffic flow and wait times	1 hour (Wed)	Optimize the agent's reward based on traffic efficiency and movement.

Week	Tasks	Expected Time	Comments
	Final Testing and Adjustments: Run tests, analyze DQN performance, and fine-tune	3 hours (Thu, Fri)	Ensure the model achieves realistic, stable results before finalizing.

References

TraCI Commands

- Control-related commands: perform a simulation step, close the connection, reload the simulation.
- Generic Parameters

For the following APIs, the ID is equal to the ID defined in sumo's input files. Here, you find their general structure.

Value Retrieval

- Traffic Objects
  - Vehicle Value Retrieval retrieve information about vehicles
  - Person Value Retrieval retrieve information about persons
  - Vehicle Type Value Retrieval retrieve information about vehicle types
  - Route Value Retrieval retrieve information about routes
- Detectors and Outputs
  - Induction Loop Value Retrieval retrieve information about induction loops
  - Lane Area Detector Value Retrieval retrieve information about lane area detectors
  - Multi-Entry-Exit Detectors Value Retrieval retrieve information about multi-entry/multi-exit detectors
  - Calibrator Value Retrieval retrieve information about calibrators
  - RouteProbe retrieve information about the RouteProbe
- Network
  - Junction Value Retrieval retrieve information about junctions
  - Edge Value Retrieval retrieve information about edges
  - Lane Value Retrieval retrieve information about lanes
- Infrastructure
  - Traffic Lights Value Retrieval retrieve information about traffic lights
  - BusStop Value Retrieval retrieve information about BusStops
  - Charging Station Value Retrieval retrieve information about charging stations
  - Parking Area Value Retrieval retrieve information about parking areas
  - Overhead Wire Value Retrieval retrieve information about overhead wires
  - Rerouter retrieve information about the rerouter
- Misc
  - Simulation Value Retrieval retrieve information about the simulation
  - GUI Value Retrieval retrieve information about the simulation visualization
  - PoI Value Retrieval retrieve information about points-of-interest
  - Polygon Value Retrieval retrieve information about polygons

State Changing

- Traffic Objects
  - Change Vehicle State change a vehicle's state
  - Change Person State change a persons state
  - Change Vehicle Type State change a vehicle type's state
  - Change Route State change a route's state
- Detectors and Outputs
  - Change Calibrator State change a calibrator state
  - Change Inductionloop State change a inductionloop state
  - Change Lane Area Detector State change a lane area detector state
- Network
  - Change Edge State change an edge's state
  - Change Lane State change a lane's state
- Infrastructure
  - Change Traffic Lights State change a traffic lights' state
  - Change Charging Station State change a charging stations's attributes
- Misc
  - Change Simulation State change the simulation
  - Change GUI State change the simulation visualization
  - Change PoI State change a point-of-interest's state (or add/remove one)
  - Change Polygon State change a polygon's state (or add/remove one)

TraCi to manipulate the traffic stuff
https://sumo.dlr.de/docs/TraCI.html

We would be needing some stuff from this like

1. Vehicle value retrieval
2. Person value retrieval
3. Route Value retrieval
4. Junction Value Retrieval
5. Edge Value Retrieval
6. Lane Value Retrieval
7. Parking Value (optional)
8. Simulation Value (This might be useful)

Some basic notes from GPT in understanding the files and the methods of TraCi that would be useful

In SUMO, different XML files serve specific purposes to define and simulate a road network. Here's a breakdown of the main file types and how to use them to analyze traffic load with TraCI.

SUMO XML File Types

1. `nod.xml` (Nodes File):
   - Defines intersections and road junctions within the simulation.
   - Each node represents a specific point in the network, typically identified by an ID and coordinates.
   - Used to establish the layout of intersections and endpoints in the network.
   - Example usage: Used in network generation, as nodes are required to define roads between them.
2. `edg.xml` (Edges File):
   - Defines roads or segments between nodes (junctions).
   - Each edge connects two nodes and has properties like speed limits, lane count, and type.
   - Essential for setting up the road layout in SUMO.
   - Example usage: Controls road properties like speed, which influences vehicle movement and network flow.
3. `net.xml` (Network File):
   - A compiled file that includes nodes, edges, connections, and traffic light information.
   - Generated by SUMO's `netconvert` tool, which converts `nod.xml`, `edg.xml`, and `con.xml` (connections file) into a single network.
   - Used as the main input for SUMO simulations.
   - Example usage: Direct input for defining the road network in SUMO; contains all road segments, intersections, and control points.

4. `rou.xml` (Routes File):
    - Defines the routes for vehicles, specifying their origins, destinations, and routes through the network.
    - Includes details about vehicle IDs, departure times, and specific paths.
    - Crucial for controlling vehicle flow in simulations.
    - Example usage: Specifies when and where vehicles enter and leave the network, affecting traffic density.

Key TraCI Functions to Monitor Traffic Load on Signal Agents

To analyze traffic load on traffic signal agents, you can monitor various factors with TraCI. Here's a list of important metrics and corresponding TraCI functions:

1. **Vehicle Count & Density**:
    - `traci.edge.getLastStepVehicleNumber(edgeID)`: Returns the number of vehicles currently on the specified edge.
    - `traci.lane.getLastStepVehicleNumber(laneID)`: Returns the number of vehicles on a specific lane.
    - Useful for monitoring the number of vehicles approaching the traffic signal.
2. **Queue Length**:
    - `traci.edge.getLastStepHaltingNumber(edgeID)`: Returns the count of vehicles in a queue on the specified edge.
    - `traci.lane.getLastStepHaltingNumber(laneID)`: Similar to the edge function, but specific to a lane.
    - Provides an estimate of congestion, as longer queues indicate higher traffic loads.
3. **Average Waiting Time**:
    - `traci.edge.getWaitingTime(edgeID)`: Returns the total waiting time for all vehicles on the edge.
    - `traci.lane.getWaitingTime(laneID)`: Similar to edge waiting time but for individual lanes.
    - Useful to assess traffic flow efficiency and identify congestion hotspots.
4. **Travel Time**:
    - `traci.edge.getTraveltime(edgeID)`: Returns the average travel time for vehicles on the edge.
    - Helps to measure delays and can be used to analyze traffic signal performance over time.
5. **Traffic Signal State**:
    - `traci.trafficlight.getPhase(tlsID)`: Returns the current phase (e.g., red, green) of the traffic light at a specified junction.
    - `traci.trafficlight.getPhaseDuration(tlsID)`: Returns the duration of the current phase.

- `traci.trafficlight.getNextSwitch(tlsID)`: Returns the time until the next phase switch.
- Monitoring these allows you to understand and adjust signal timings based on real-time traffic load.

6. **Vehicle Speed**:
   - `traci.edge.getLastStepMeanSpeed(edgeID)`: Returns the mean speed of vehicles on the edge.
   - Reduced speed can be an indicator of high traffic density or congestion.
7. **Emissions Data** (Optional):
   - `traci.vehicle.getCO2Emission(vehID)`, `traci.vehicle.getNOxEmission(vehID)`: Provides information on emissions for individual vehicles.
   - Emission data can help indirectly evaluate traffic density and flow quality

Check if these methods are good enough or we need to defgine it custom way
https://github.com/LucasAlegre/sumo-rl/blob/main/sumo_rl/environment/traffic_signal.py

Documentation (better one)
https://lucasalegre.github.io/sumo-rl/documentation/traffic_signal/