

UE18CS390B - Capstone Project Phase - 2

SEMESTER - VII

END SEMESTER ASSESSMENT

Project Title : Smart Traffic Light Controller using Deep Reinforcement Learning

Project ID : PW22NKS01

Project Guide : Prof. Nagegowda K S

Project Team : Abhishek A, Krishna P Hegde, Prathvik Nayak, A Lakshmi Prasad

Outline

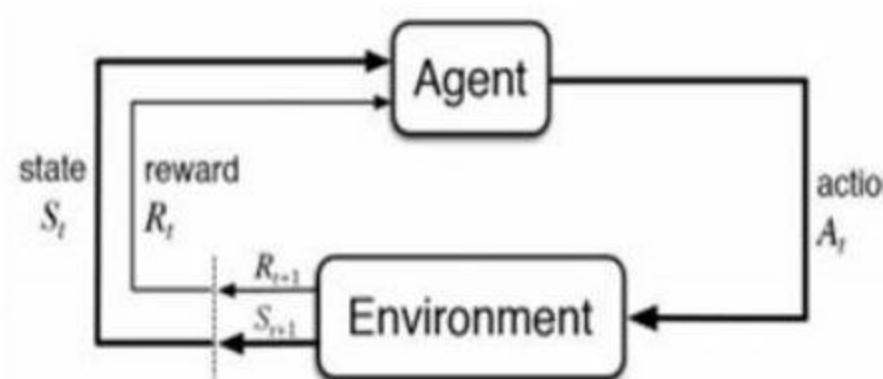
- Abstract
- Team Roles and Responsibilities.
- Summary of Requirements and Design (Capstone Phase - 1)
- Summary of Methodology / Approach (Capstone Phase - 1)
- Design Description
- Modules and Implementation Details
- Project Demonstration and Walkthrough
- Test Plan and Strategy
- Results and Discussion
- Lessons Learnt
- Conclusion and Future Work
- References

Abstract

- Conventional traffic signals use a regular timer based approach which do not handle dynamic traffic conditions.
- Due to this, the vehicles have to wait for a long time span even if the traffic density is less.
- In order to improve this we present a Reinforcement Learning approach to control traffic signals so as to reduce traffic congestion in crossroads and intersections.
- The scope of this project is to initially simulate traffic in a 4 way intersection and then extend it for multiple intersections using a traffic simulator(SUMO).

Abstract

- We use an agent(traffic lights system) to perceive the environment and train this agent to self learn using Reinforcement learning.
- In order to design a system based on the Reinforcement learning approach, we define the state representation, the action set, the reward function and the agent learning techniques involved.



Abstract

- The learning mechanism used here is Deep Q-Learning, which is a combination of two aspects widely adopted in the field of reinforcement learning which are deep neural networks and Q-Learning.
- It is a form of model-free reinforcement learning technique which is used for determining the value of an action that was taken is Q-learning.
- Q in Q-learning represents ‘quality’. This method involves assigning a Q-value or a ‘quality value’ to an action that was taken to alter the environment’s state.
- This Q value is obtained from a function known as Q-function. We will be training our neural network to learn the optimal Q function to achieve the best results.

Abstract

- Reinforcement learning has a lot of applications in real world.
- Some of the applications are:
 - Auto driving cars.
 - Robotics for industrial automation.
 - Business strategy planning.
 - Machine learning and data processing.
 - Automated medical diagnostics.
 - Aircraft control and robot motion control.
 - Video games

Team Roles and Responsibilities



Name	Tasks/Modules
Abhishek A	traffic_generation, model creation, literature survey, training, results plotting and visualization, reports, IEEE paper.
Krishna P Hegde	traffic_generation, model creation, literature survey, training, GUI, reports, IEEE paper.
Prathvik Nayak	Road network creation in netedit, literature survey, model creation, training, SUMO testing, report, IEEE paper.
A Lakshmi Prasad	Literature survey, training, SUMO testing, report, IEEE paper.

Summary of Requirements and Design



Requirements

- Generation of road network and flow network in SUMO simulator and creating simulations.
- The software should simulate heavy traffic for training the agent.
- Training the agent (traffic controller) with the simulations.
- Testing the agent with new simulations.

Summary of Requirements and Design



- The agent must be able to minimize traffic congestion as much as possible to allow smooth flow of traffic.
- Analyzing the reward trends and agent performance while training.
- Performance analysis of the smart traffic controller when compared to conventional traffic signals.

Summary of Requirements and Design



Constraints/Dependencies/Assumptions/Risks

- This simulation is done on SUMO software.
- Since this is a simulation, it may not be very precise compared to real world traffic, however it is still capable to simulate traffic in as realistic manner as possible.
- The TraCI interface is used to interact with SUMO using python.
- Training the agent could be a very tedious job as we will need to run a large number of simulations which is time consuming.

Summary of Requirements and Design

- Extending to multiple intersections could be challenging and complex as we will need to train each intersection in the network at the same time equally.
- Further implementation in real world could be a complex process due to increase in resources such as sensors to scan the traffic.
- The sensors when implemented in the real world should be able to correctly produce the inputs so that the model can produce better outputs.

Summary of Methodology / Approach



- Reinforcement learning is the concept that we will be using to solve this problem of traffic congestion and traffic control.
- Here we have an agent perceiving the environment and depending on that state of the environment, it performs a certain action. If the action taken was a good one, it gets a positive reward else it gets a negative reward. The agent has a goal which it achieves by maximizing the rewards.
- The learning mechanism used here is deep Q learning, that is a mix of DNN and Q-learning. We propose to use an experience replay mechanism for enhancing the performance and efficiency of the agent. We also use an epsilon greedy policy for training the agent.

Summary of Methodology / Approach

Action set

- The possible actions that can be taken by the agent are contained in the action set.
- The number of actions in the action set is the same as the number of incoming lanes in an intersection.
- For a 4-way intersection there are 4 possible actions and each action is a phase of the traffic signal that is North Advance, East Advance, South Advance and West Advance.
- Similarity in 3-way will have 3 actions in its action set and 5-way will have 5 actions in its action set.

Summary of Methodology / Approach

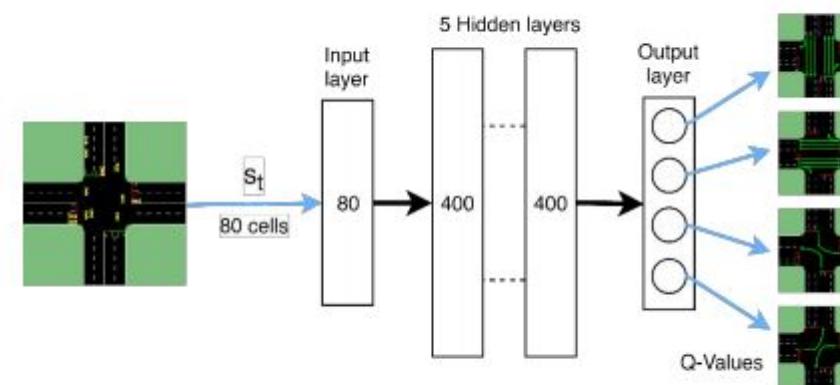


Reward

- Reward is something which the agent receives as a consequence of its chosen action.
- A good action results in a positive reward while a bad action results in a negative reward.
- We have used change in cumulative waiting time between actions as the metric to calculate the reward.

Summary of Methodology / Approach

- A DTSE (Discrete traffic state encoding) methodology is used for getting the state information from the environment.
- Here, each lane in the environment is discretized into cells.
- This forms a matrix containing 0s and 1s wherein 1 represents the presence of vehicle in that particular cell and 0 represents the absence of vehicle in that particular cell.
- This matrix is passed as an input to the neural network which gives out Q values for each action as an output.



Summary of Methodology / Approach

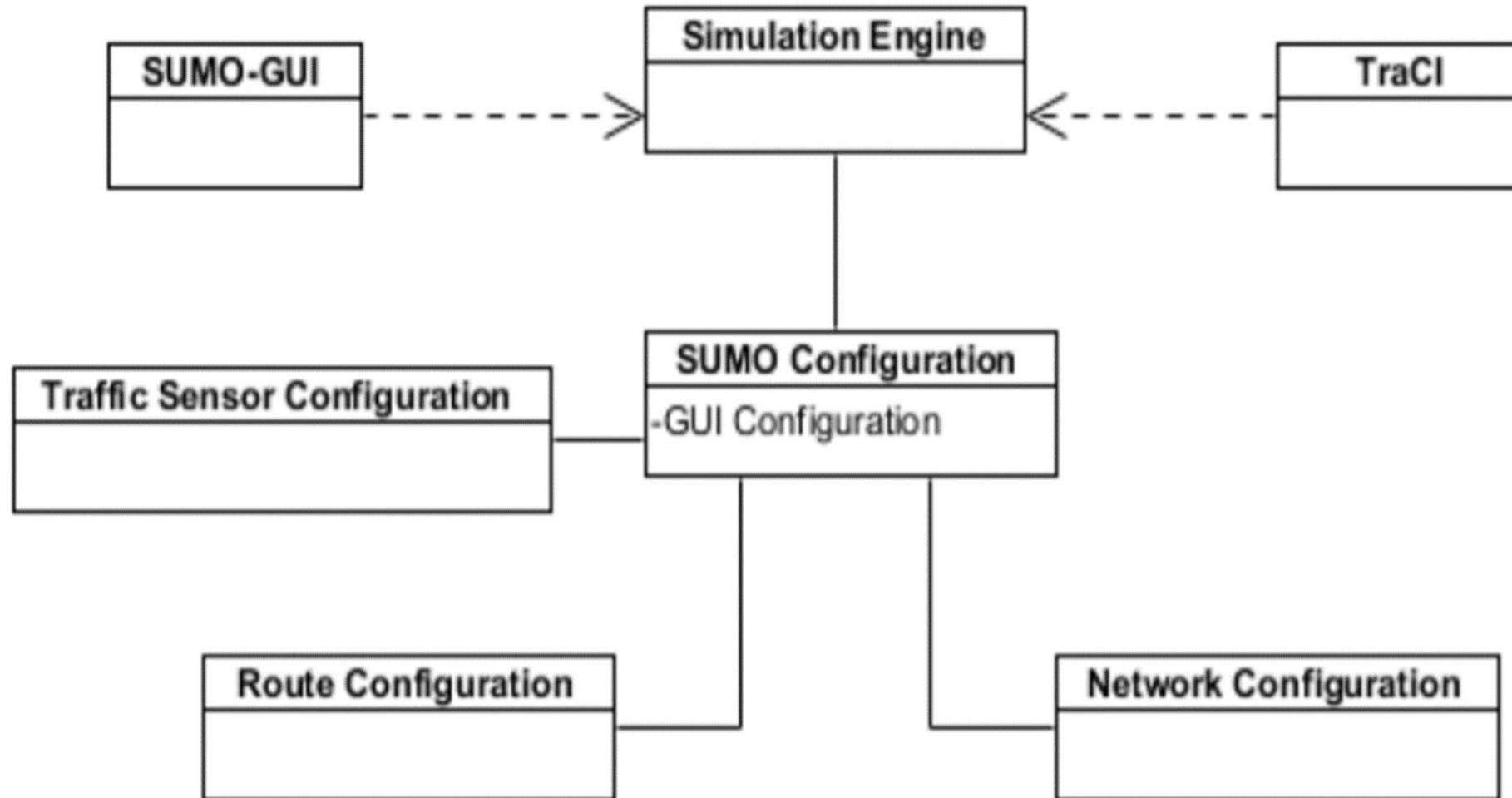
- The equation for Q-learning function is

$$Q(s_t, a_t) = \text{reward} + \gamma \cdot \max_A Q'(s_{t+1}, a_{t+1})$$

- The state information captured during DTSE is stored in memory and a group of samples is retrieved from the memory and trained as a batch.
- Here we make use of a function approximators (neural network models) to the optimal Q-function.
- The epsilon greedy policy decides whether an agent performs an explorative action or an exploitative action.

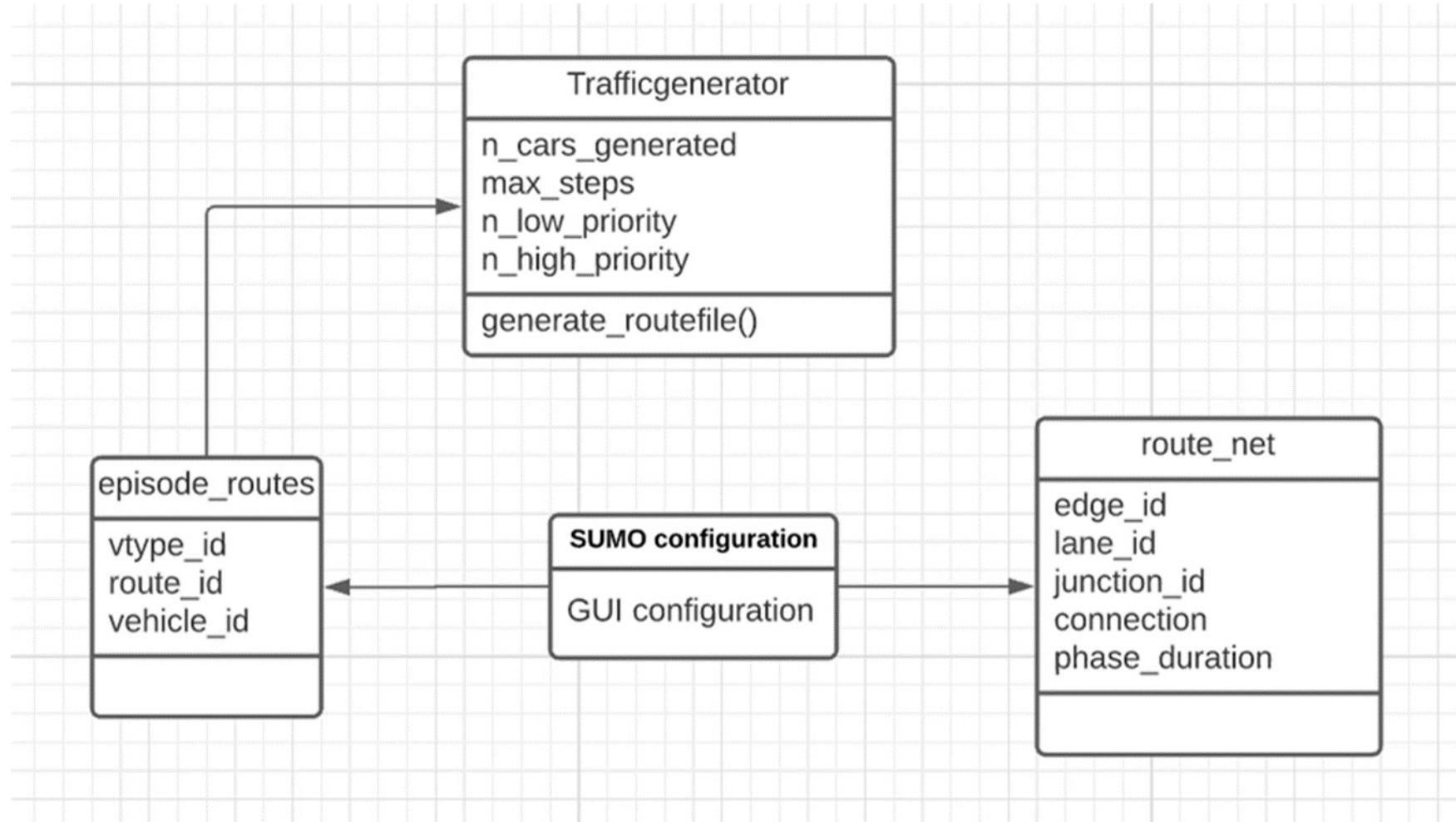
Design Description

1. Master Class Diagram



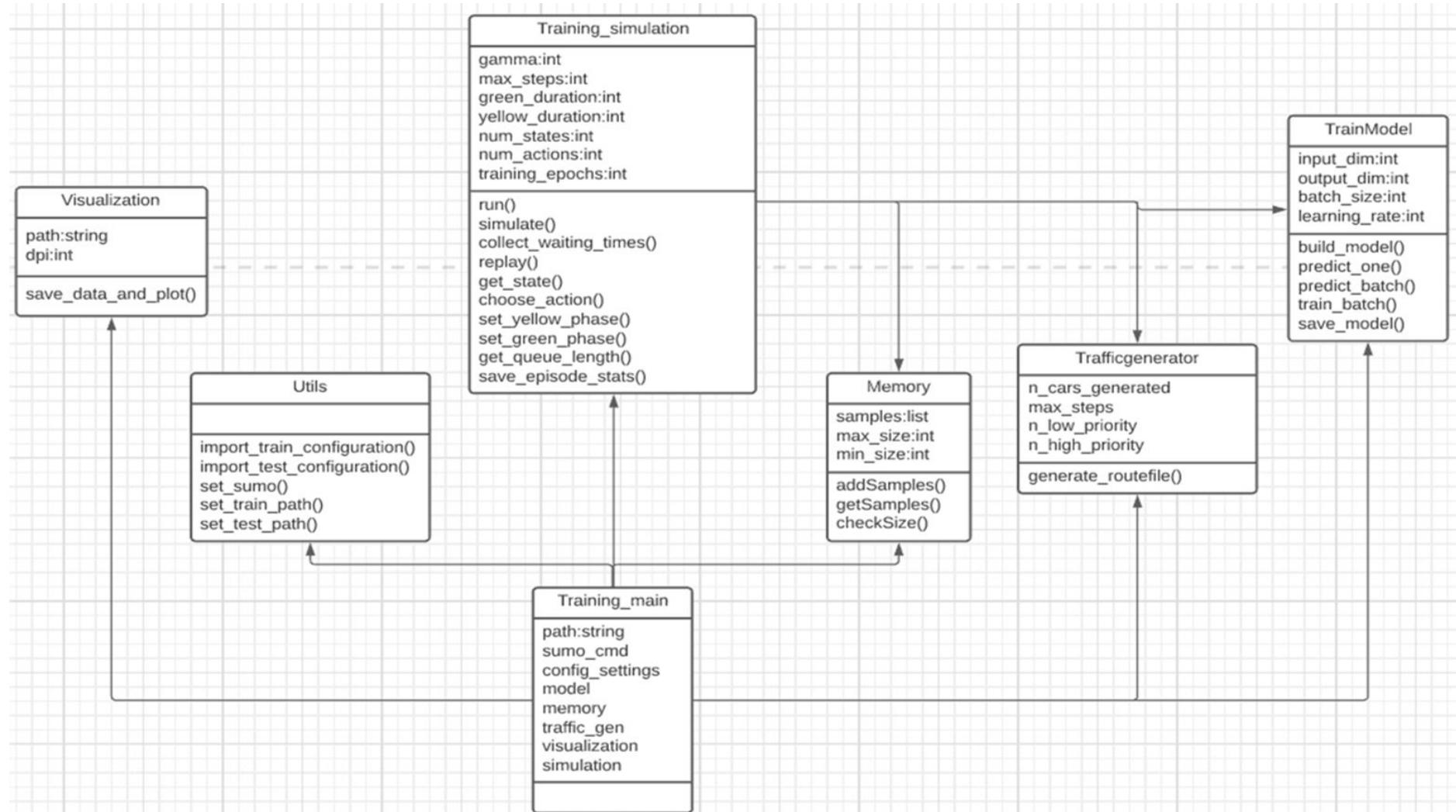
Design Description

2. Traffic generation class diagram



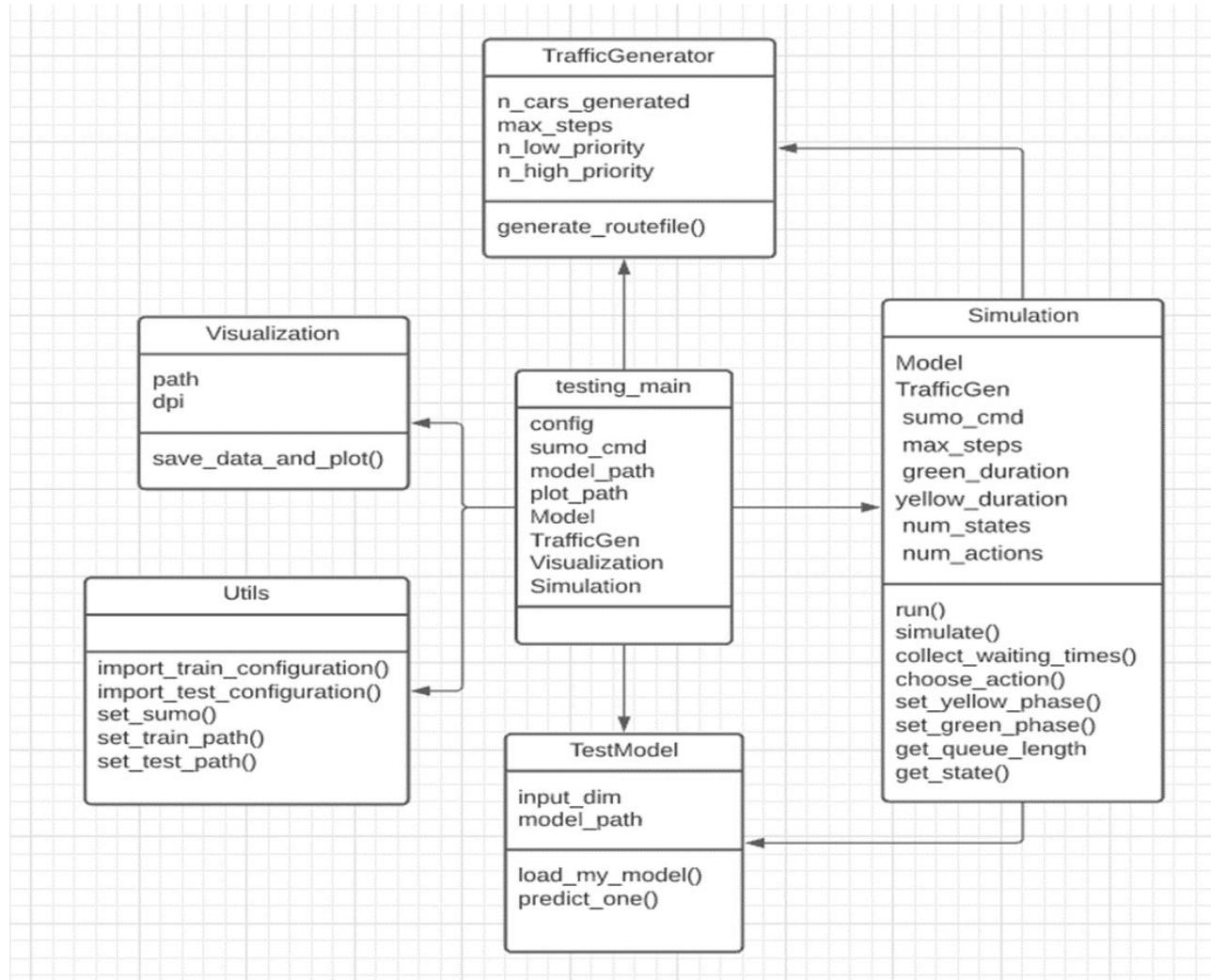
Design Description

3. Training and simulation class diagram



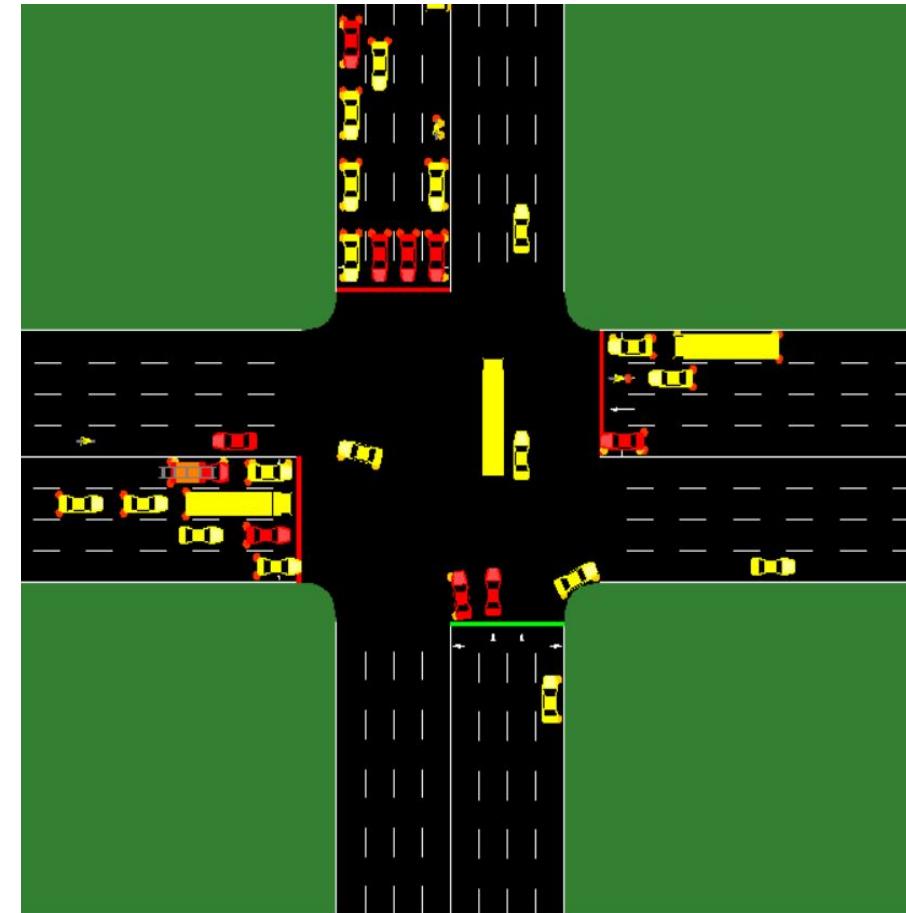
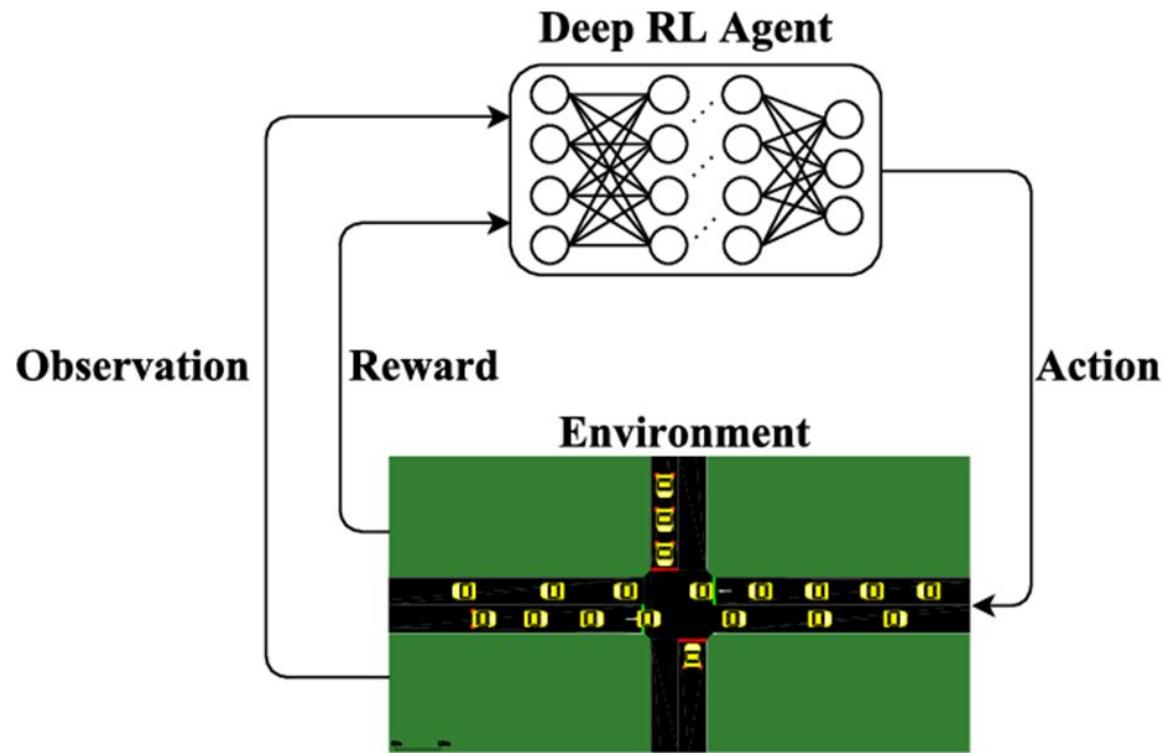
Design Description

4. Testing and simulation class diagram



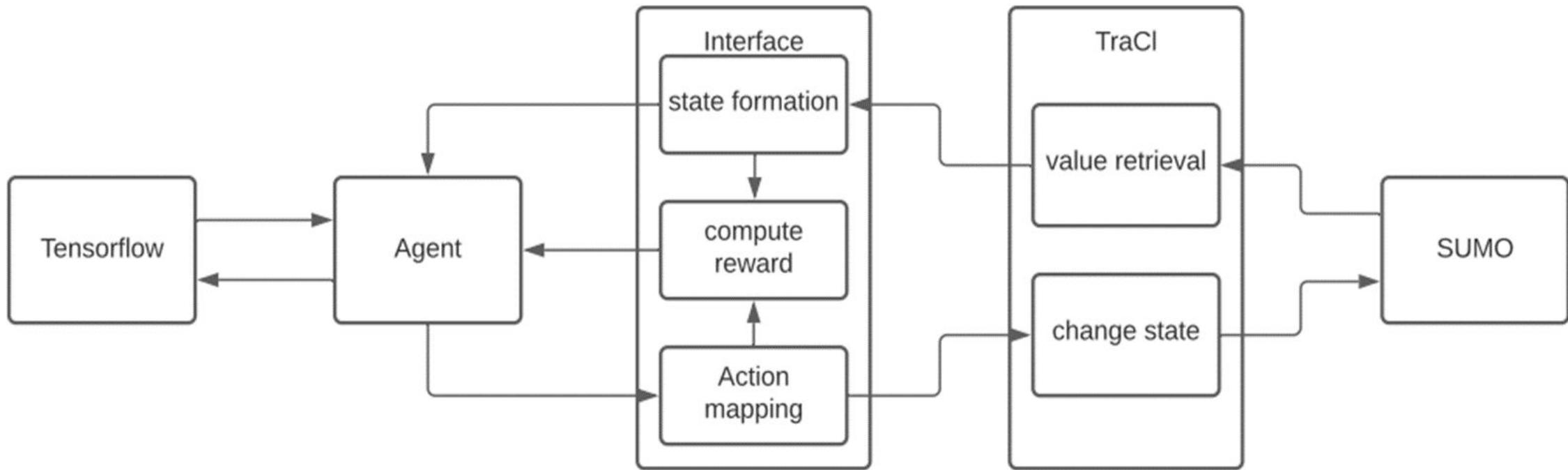
Design Description

4. RL Framework and SUMO interface



Design Description

5. External Interface Diagram



Modules and Implementation Details



Traffic generation

- The road network structure(environment.net.xml) and the initial traffic light sequence with durations for both single and double intersection environment was created using netedit software.
- The traffic on these networks was generated using a python script(generator.py).
- The generator.py file takes the parameter ‘no of cars per episode’ and generates traffic with all kinds of vehicles. You can even pass no of high priority and low priority vehicles.
- The generation of vehicles is distributed according to Weibull distribution.

Libraries/Technologies used - numpy, math

Modules and Implementation Details

Model creation

- We will be creating two models.
- The first model consisting of a fully connected ANN with m neurons in the input layer, 5 hidden layers with 400 neurons each with relu activation function and an output layer of n neurons with a linear activation function and an Adam optimizer. (Here m is the I/P dimension and n is the O/P dimension)
- The second model uses a CNN with 2 input matrices (position and velocity). The first layer of convolution consists of 16 filters of 4×4 with stride=2, second layer of convolution consists of 32 filters of size 2×2 with stride 1, the 3rd and 4th layer are fully connected with size 128 and 64 respectively and the output layer is a linear layer with size n . (Here n is the output dimension)

Libraries used - tensorflow, numpy, keras, sys, os

Modules and Implementation Details

Simulation and training

- Here we'll be making use of traci api to connect to the simulations.
 - From the simulations we will be retrieving multiple state spaces using the DTSE method and store them in memory.
 - We will discretize the incoming lanes into cells to identify the presence or absence of vehicles in them.
 - We will be training the model using an experience replay method to improve its efficiency and reduce correlation between consecutive simulation steps.
 - The model will also be trained to select an action using the epsilon greedy policy.
 - For multiple intersections we will be training 2 separate models in the same simulation.
- Libraries used - traci, numpy, random, timeit, os

Modules and Implementation Details



Memory

- We are implementing the experience replay mechanism.
- The memory.py file will handle the memorization for the experience replay mechanism.
- A function adds a sample into the memory, while another function retrieves a batch of samples from the memory.

Libraries used - random

Utils

- This file contains directory related functions such as loading models for testing and creating different versions of trained models, importing configuration settings, etc.

Libraries used - configparser, sumolib, sys, os, checkBinary

Modules and Implementation Details



Testing

- We tested the model with a single test episode running a completely different and random simulation with emergency vehicles configured in such a way that they can ignore the red lights.
- We tested both the models(ANN and CNN) for multiple intersections as well.

Libraries/modules used - traffic generator, simulation, model, visualization, utils, os.

Visualization and analysis study

- Once the testing is done we performed analysis study to see the performance of the agent in both the models compared to normal traffic, to study the reward trend, and study the effects of undertraining in complex networks.

Libraries used - matplotlib, os

Walkthrough



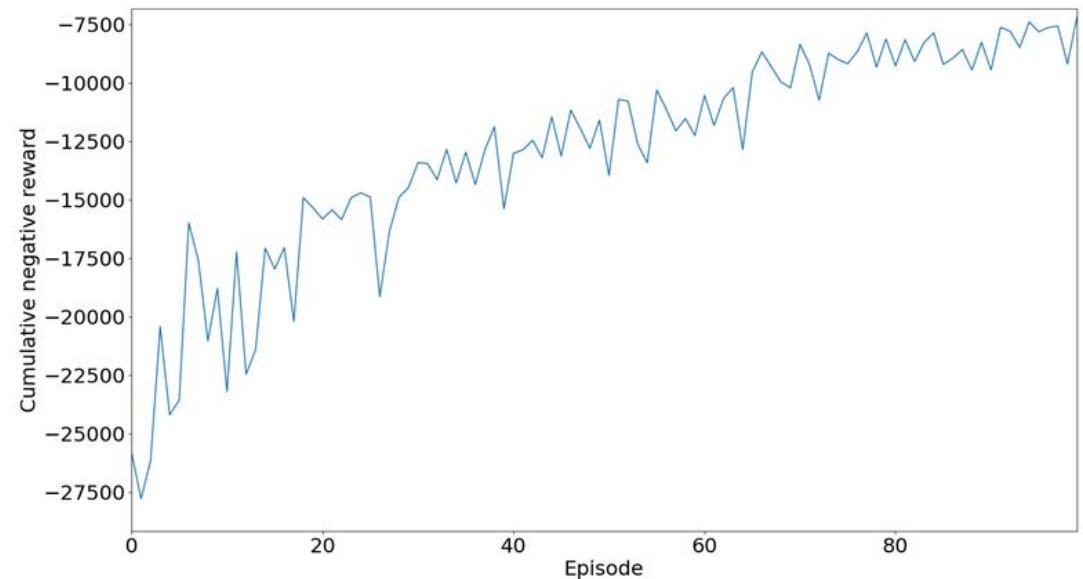
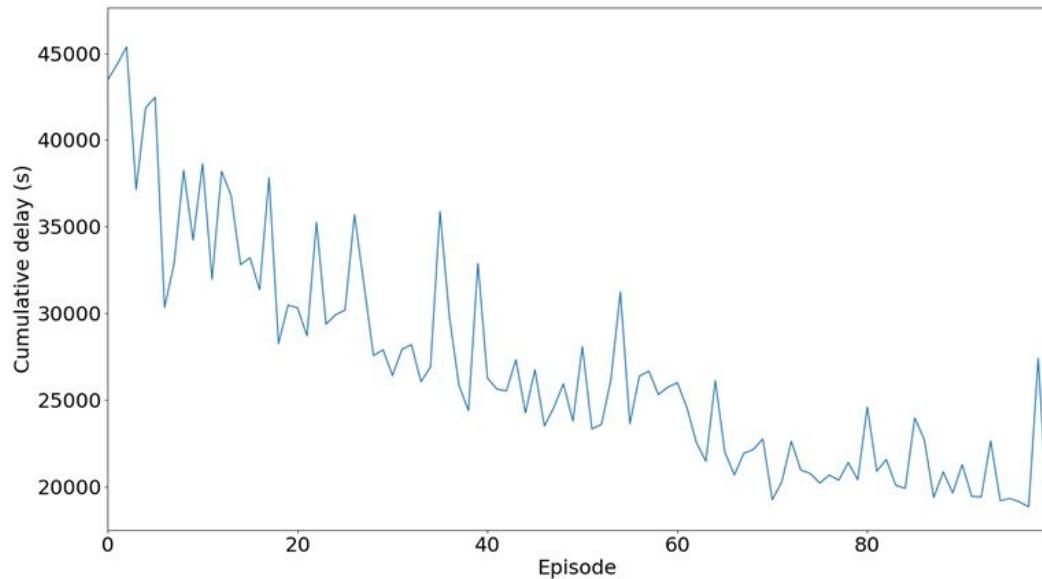
Demo and Product walk through

Test Plan and Strategy

- Testing activities were carried out on every trained model.
- Each trained model was tested with a single test episode by running a random simulation.
- Data such as queue length, waiting time of all the vehicles and reward was collected during the test simulation.
- Collected data was used to plot graphs to interpret results.

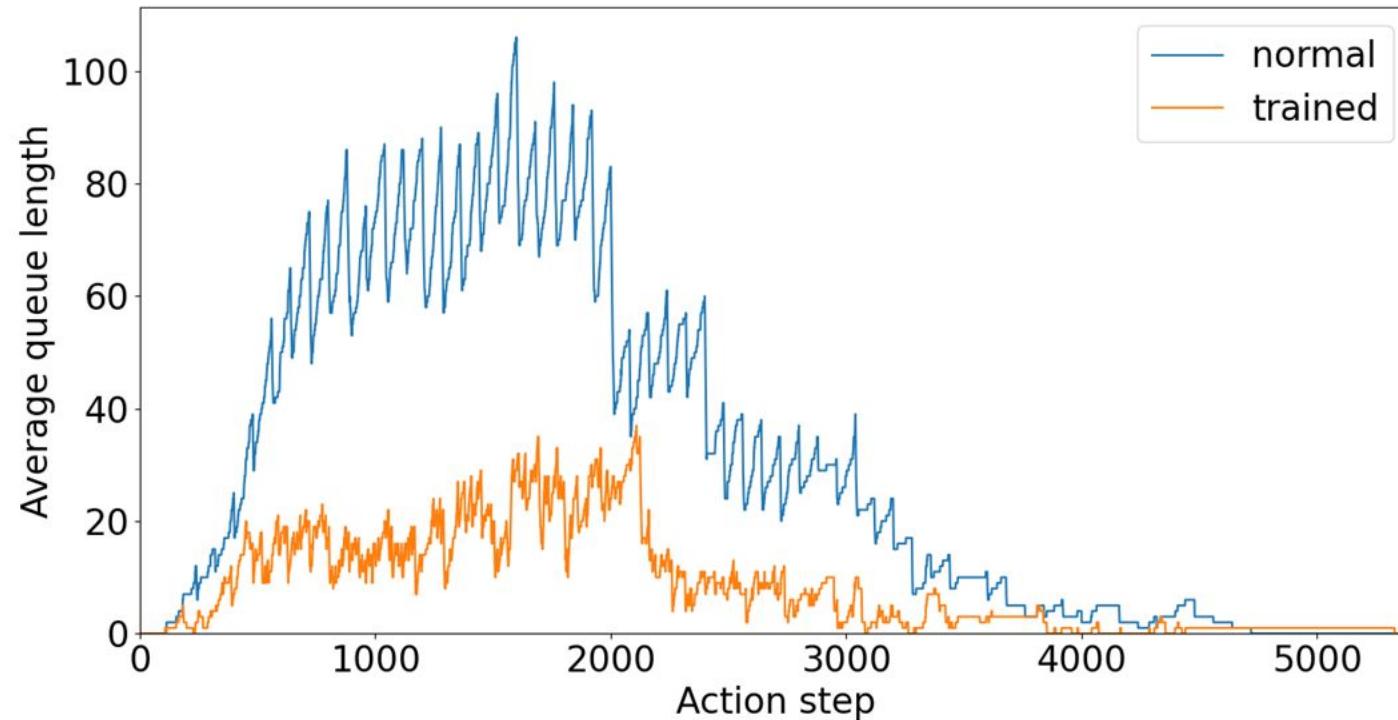
Results and Discussion

- The trained model performed much better than the normal simulation, as the trained model had learnt an optimal policy function to maximize the rewards and choose an optimal action.
- Our agents were able to reduce the cumulative delay of vehicles and maximize the rewards during the training process as we can see in the below graphs.



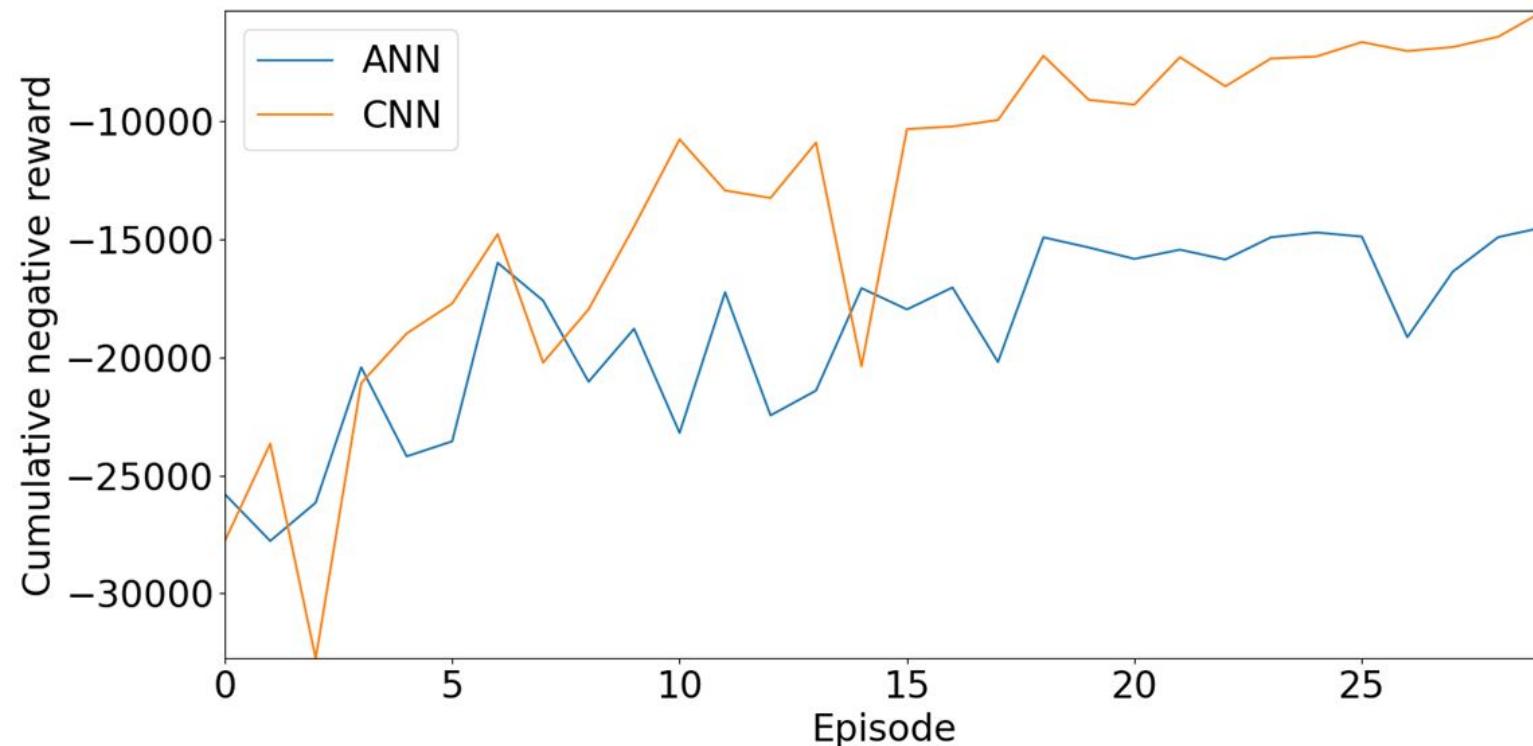
Results and Discussion

- On comparison with traditional traffic signals we can observe that the trained agent was able to reduce queuing in traffic signals significantly as seen in the below graph.



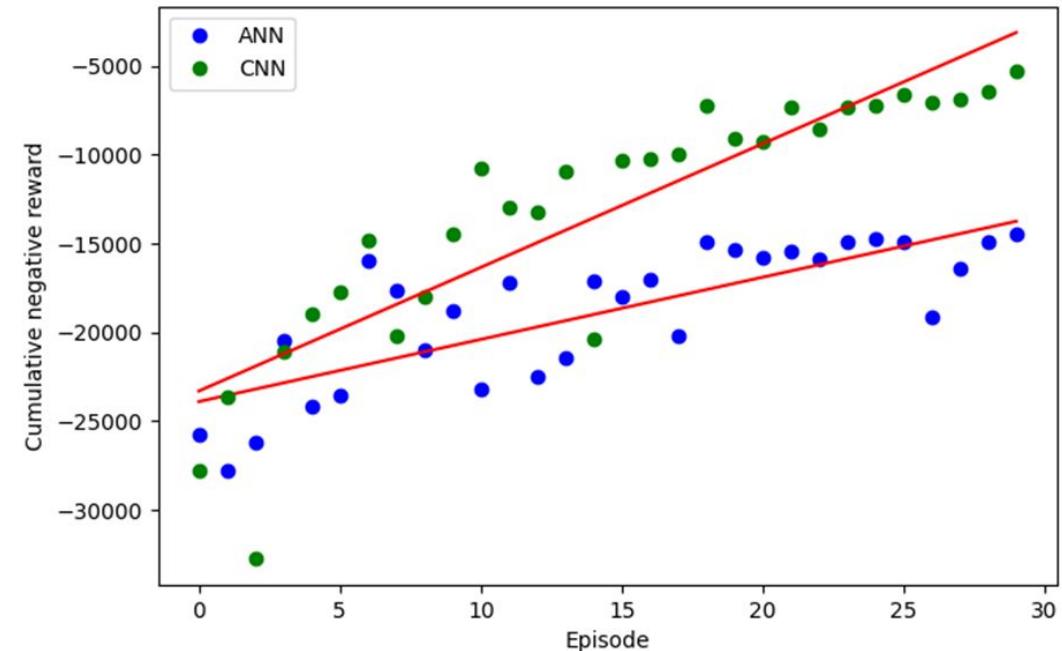
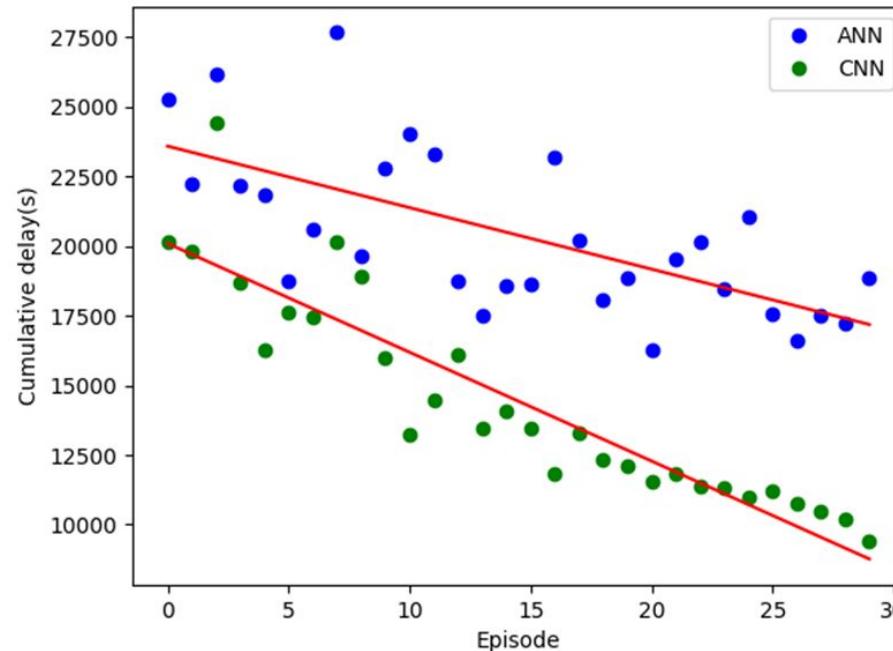
Results and Discussion

- The agents trained using the convolutional neural networks [CNN] performed much better than the agents trained using the Artificial neural network [ANN] as seen in the plots below.



Results and Discussion

- On analysis of the reward trend we can see that the CNN is able to maximise the rewards much quicker compared to ANN with lesser training, we can see that the CNN has a much more negative slope than ANN while reducing the cumulative delay during the training phase.

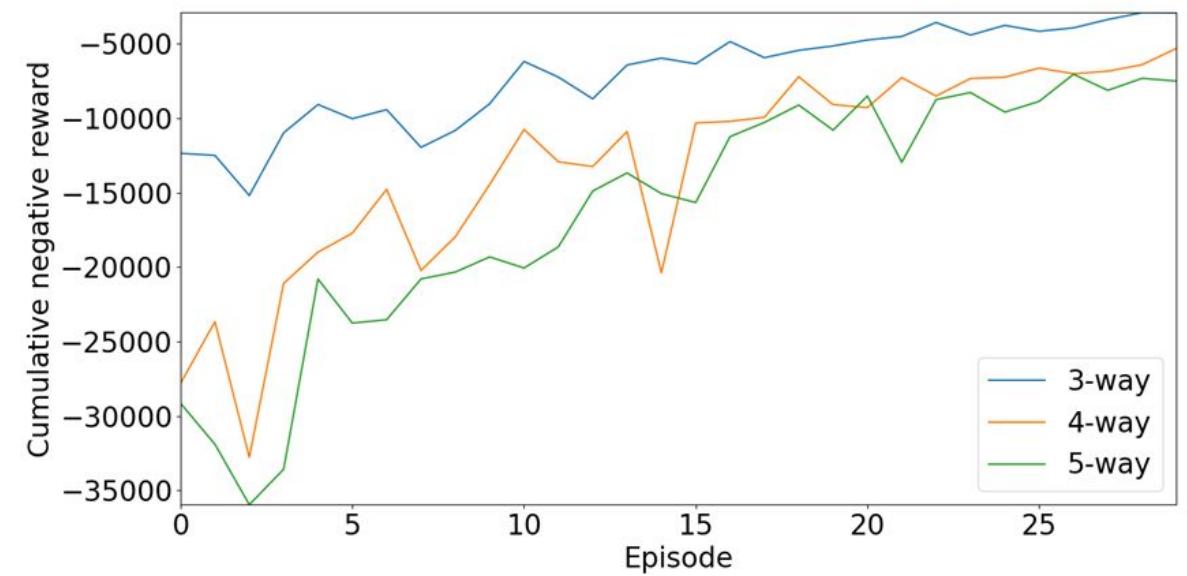
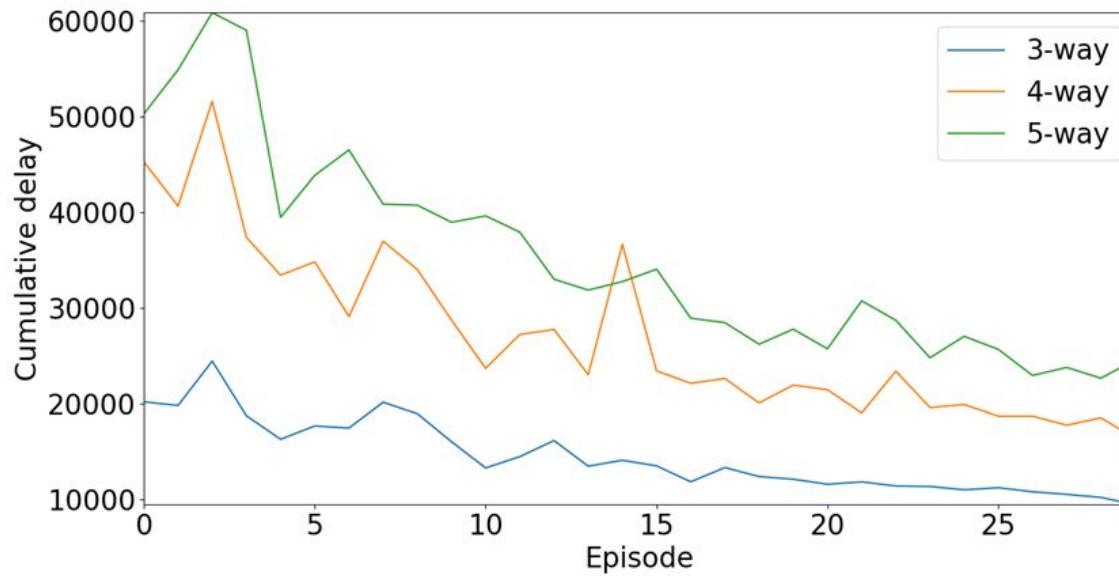


Results and Discussion

- The reason why CNN performs much better compared to ANN is because of its ability to extract important features using DTSE.
- This allows CNN to develop high-level state representations. Moreover the CNN takes multiple data inputs, in our case the position and velocity matrix compared to the position matrix alone in the ANN.
- This allows CNNs to achieve better results with higher accuracy. Although CNN takes longer to train, it gives efficient results much quicker.

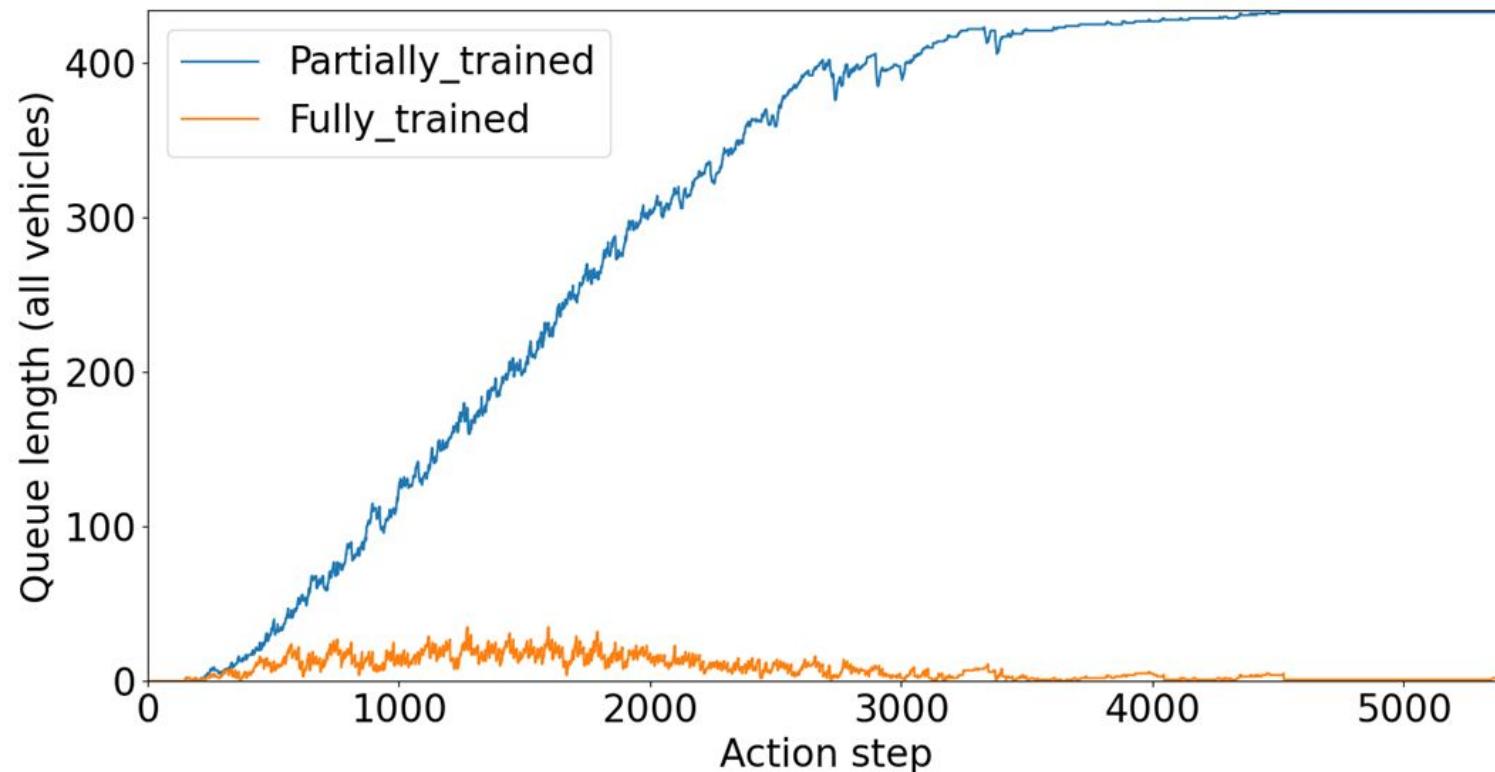
Results and Discussion

- It was also seen that the models performed better in smaller intersections than the bigger ones.



Results and Discussion

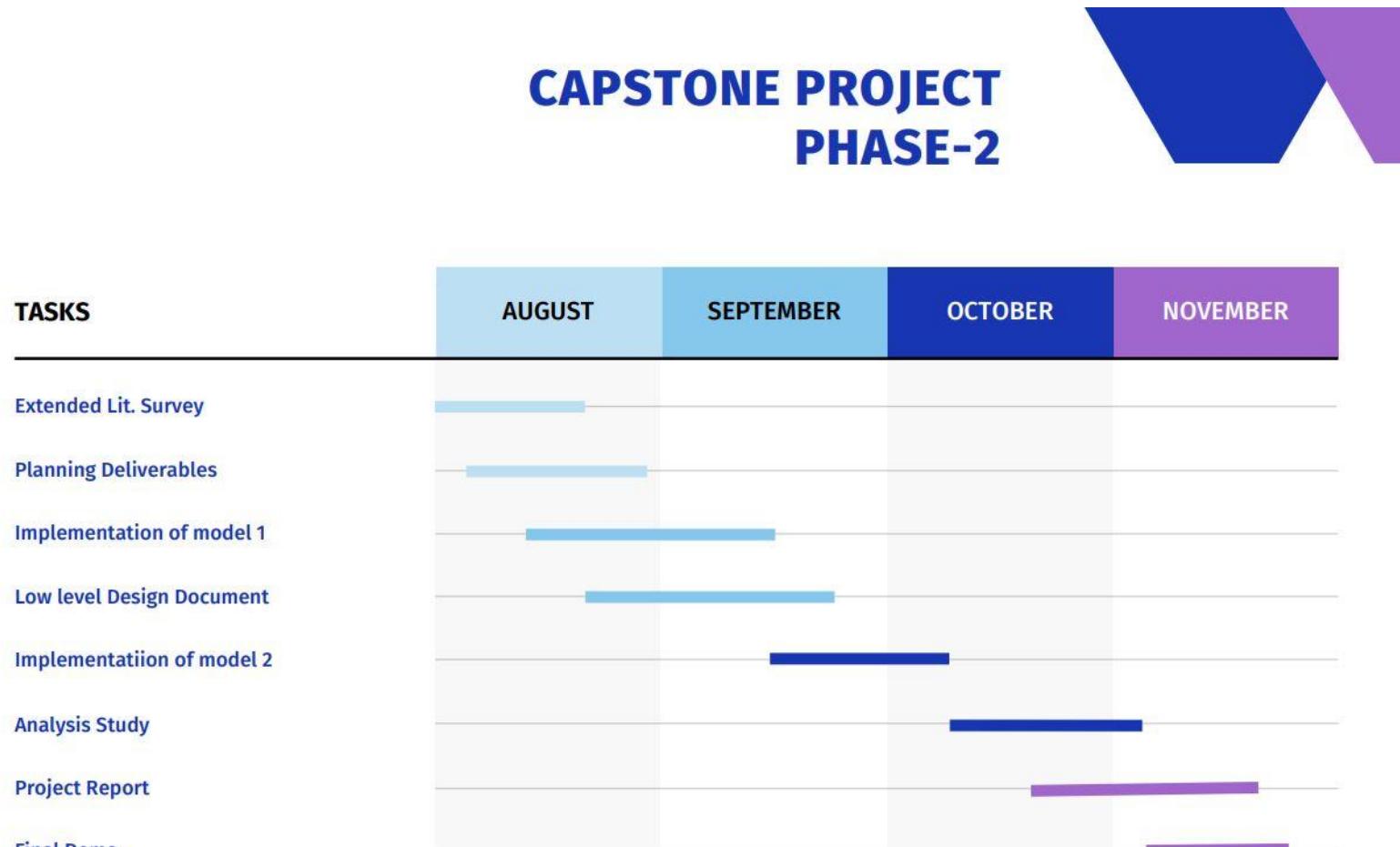
- For complex intersection networks, when one model is undertrained, we found that it performs worse when compared to the fully trained models.



Results and Discussion

- From the previous graph we can see that while training for complex networks, each model has to be trained fully to achieve better results.
- Even if one of them is undertrained, the whole network will perform worse, since each individual network is interconnected and queue_length in one network impacts the whole network

Schedule



- The project was implemented according to the schedule as planned.
- All the planned tasks were completed within the timeline.
- The planned schedule is shown in the form of a gantt chart on the left.

Lessons Learnt



- RL approach made it easier to solve this problem, however there were other training techniques within RL which could have been implemented but due to constraints on resources, computing power and time it couldn't be implemented.
- Proper use of Machine learning approaches can provide effective and excellent results.
- Reinforcement learning, despite its limited application in the real world, is still one of the hot topics in the field of Artificial Intelligence and could potentially have a great impact in the future.

Lessons Learnt

- The main issue that has been overcome is traffic congestion and reducing the cumulative waiting time of vehicles in traffic intersections using Reinforcement learning approach.
- Issues faced during implementation include:
 - ❖ Increase in complexity when moving from 3-way to 4-way to 5-way and then to complex intersections.
 - ❖ Deciding on what values do be used for the hyperparameters of the neural networks.
 - ❖ Deciding on what metrics to be considered for analysis and result interpretation.
 - ❖ Training the model takes a lot of time.

Lessons Learnt

- The novelty of our project was to implement a smart traffic controller using Reinforcement learning to efficiently manage and control traffic congestions and to try and implement it for multiple intersections in a complex networks.
- We have tried to improve the efficiency and performance of the agents by choosing appropriate models and correct hyperparameters to get the best results.
- We have also performed a lot of analysis during training as well as testing to understand where and how we can improve the performance of the agent, and also an observation was done to see the effects of undertraining an agent in complex intersections.

Lessons Learnt



- We have also tried to take into consideration the emergency vehicles in the simulation and tried to design our agent in such a way that it would help in reducing both the traffic congestion as well as forming a rescue lane for emergency vehicles in the simulation.
- However this problem has its limitations since we are using a simulator, but it could be solved using other techniques on top of the existing RL technique to implement it in the real world in the near future.
- An interpretation of the results showed that RL approach was very efficient compared to traditional methods.

Lessons Learnt



- However there are various challenges of using Reinforcement learning
 - Feature/reward design which should be very involved
 - Parameters may affect the speed of learning.
 - Realistic environments can have partial observability.
 - Too much Reinforcement may lead to an overload of states which can diminish the results.
 - Realistic environments can be non-stationary.

Conclusion and Future work

- The work carried out presented a RL approach to monitor and control traffic lights in a much more effective manner wherein they are able to adjust dynamically to real time traffic and take appropriate actions.
- This work was carried out on a traffic simulator which provides a realistic environment to carry out complex simulations.
- The agents were trained using the DQN mechanism and its performance was analysed using various parameters.

Conclusion and Future work

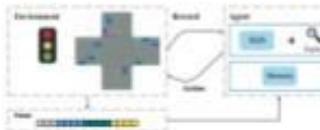
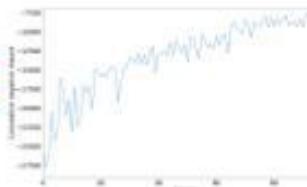
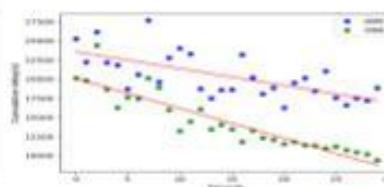
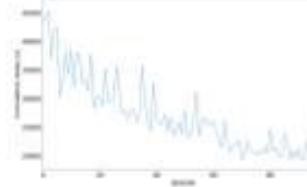
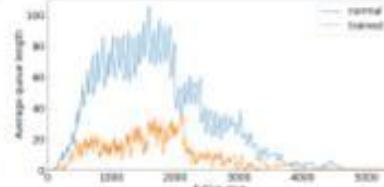
- There are however many things that can be improved in this work.
- Future works could include improving the results achieved and expanding it to a much larger network.
- The training of the agents could be improved by using different techniques such as target network, clipping rewards and skipping frames.
- We can also try using different reward and Q functions to try and improve the model further. Proper analysis is important to understand the advantages and possible disadvantages that it can bring on introduction to the real world.

Poster



Smart Traffic Light Controller Using Deep Reinforcement Learning

Krishna P Hegde, Abhishek A, Prathvik Nayak, A Lakshmi Prasad & Prof. Nagegowda K S
Department Of Computer Science and Engineering, PES University.

Problem Statement	Design Approaches / Methods	Summary of Project Outcome
<p>Smart Traffic Light Controller which efficiently and dynamically manages the congestion in the networks using Deep Reinforcement Learning. The implementation of the project is done on a simulation platform called SUMO.</p>	<p>The approach is to use DTSE(discrete traffic state encoding). The agent(traffic light) observes the state of the environment and in each arm of the intersection. The incoming lanes are discretized into cells that can identify the presence or absence of a vehicle inside them And then it trains them accordingly to maximise reward and reduce queue length and delay.</p>	<p>The project consists of different forms of networks like 3-way, 4-way, 5-way, double and complex intersections which are trained with ANN and CNN models. The outcome of the project is as expected with the initial estimate.</p>
Background	<h3>Results and Discussion</h3> <p>Below graphs (Left to right) shows that 1) cumulative negative reward increases as the episodes proceed. 2) the comparison between ANN and CNN model shows that CNN performed better. 3) the cumulative delay decreases drastically as the number of episodes increases. 4) the comparison between the untrained model(Traditional traffic) and the trained one with respect to average queue length and the action step.</p>	
<p>The research that has been conducted on traffic signal control using reinforcement learning has been significant. Earlier efforts were limited by simple simulations and a lack of computational power.</p> 	   	
Project Requirements / Product Features	<h3>Conclusions and Future Work</h3> <p>An RL approach was able to monitor and control traffic lights in a much more effective manner where appropriate actions were taken on the agents to adjust dynamically to real time traffic. Future works could include improving the results achieved and expanding it to a much larger network. Introduction to the real world needs proper analysing with different test cases.</p> <h3>References</h3> <ol style="list-style-type: none">1. Andrea Vidali, Luca Crocianni, Giuseppe Vizzari, Stefania Bandini, "A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management".2. Wade Genders, Saiedeh Razavi, "Using a Deep Reinforcement Learning Agent for Traffic Signal Control".	
<p>List of requirements. Generation of road networks in SUMO simulator. Testing the agents with new simulations for better performance. Product features. Agent can detect the traffic from the intersection. Based on the state of the traffic, the agent takes appropriate action to minimise traffic congestion. The cumulative delay and queue length is significantly lesser when compared to the traditional traffic system. The more the agent is trained, better it performs.</p>	 <p>Dr. Nagegowda K S</p>  <p>Krishna P Hegde</p>  <p>Abhishek A</p>  <p>Prathvik Nayak</p>  <p>A Laskhmi Prasad</p>	

References

1. Andrea Vidali, Luca Crociani, Giuseppe Vizzari, Stefania Bandini, “A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management”.
2. Wade Genders , Saiedeh Razavi, “Using a Deep Reinforcement Learning Agent for Traffic Signal Control”.
3. Hua Wei, Guanjie Zheng, Huaxiu Yao, Zhenhui Li, “IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control”.
4. Arel C. Liu1 T. Urbanik, A.G. Kohls, “Reinforcement learning-based multi-agent system for network traffic signal control”
5. SUMO documentation : <https://sumo.dlr.de/docs/> Copyright © 2001-2021 German Aerospace Center (DLR) and others.

Thank
You