Traffic Flow Prediction Using Machine Learning Algorithms

Suneel Kumar[1], Ravi Shankar[2], Ankit Patel[3]

Student, Dept. of Computer Science, Lovely Professional University Phagwara, Punjab, India

---***---

Abstract - *Traffic control is the biggest problem and challenge in all over the world, in this project we tried to solve the problem with the help of machine learning algorithm to deal with traffic challenges.in this project we have used reinforcement learning for controlling traffic light and we have used artificial environment for simulation purpose which is SUMO, in we can see the vehicle in action and outcome and e can control the delay time of vehicle and can set the delay time in SUMO environment.*

1. INTRODUCTION

The main aim of monitoring traffic and its control is to minimize the congestion and we must say that this leading hot topic nowadays also like scientists want to make a home on mars and we are still dealing with traffic problems and all that, now day peoples are shifting towards urban areas. The united nation revealed that from 1950 to 2020 more than 56.2% of the global population used to live in cities, and also predicted that by 2040 about 65% of the population will be urbanized. Changes in the situation make it very hard to deal with high population and transportation. Technology is developing exponentially and autonomous driving is futuristic, for dealing with these traffic problems and their challenges. AI is the biggest tool to solve this problem, we focus on two facets: 1. Regularization of traffic, 2. situation analysis which agent-based model for simulation in this temperament of the driver and other necessary things shaped computer in the virtual environment. The last, in specific, have reached enough insolubility, and ductility and have proven ability to validate making of a decision in research of different ways to control traffic in urban. For regulation of traffic, the utility of simulator, with the help of machine learning, we created a chance for investigation of probably to recruit virtual environment for getting output with a special situation with reinforcement learning. In this we created a 4-way intersection associated with traffic lights, we have to manage via an autonomous agent to understand the traffic situation and take advantage bear out in simulating situation probably represents an appreciable traffic condition. We simulated in SUMO (simulation of urban mobility) tool provides a very good virtual environment in which we can regularize the traffic, there are so many features of sumo environment one of them is we did PO-external programming for getting desired output, we define reward for every action, controlling agent of the traffic light. In this, we work on the management of traffic lights with reinforcement learning.

2. RELATED WORK

Nowadays, machine learning and artificial intelligence play a very important role, in every field. There are many techniques available from the old days to modern to enhance the flow of traffic and rush handling. Overcrowding occurs when traffic is exceeded by the capacity of the road, hence travel time and length of queue surge, and the speed of vehicles becomes lower, specifically when the demand for traffic is very high. The solution to the congestion problem can divide into 2 phases, firstly flow of traffic management. In the 1st phase infrastructure of traffic, roleplay the rush organisation of traffic signal. Over a few decades, there is much research done to get solutions and improve the timing of signals. The 2nd phase deals with the infrastructure of the road, which plays a big role in rush management. The reinforcement model used for the last 20 years. with the advertisement in audio and video the interest in this technique increased. Deep reinforcement learning has big potential to solve traffic control as compared to other approaches. Traffic is controlled by single-agent reinforcement learning, its serialized in 2 phases. The Q-learning agent is used by pre-synchronized signals under fixed flow and varying flows. The result has been used to evolve the distinctive situation of traffic. There is another study that is based on a 1-junction with having a 3 Q learning agent. It used various state representations when vehicles arrive at the green light, red light queues, cumulative delay, and queue length. It was executed in very high traffic and it is showing a very good result. For single singled junction single-agent reinforcement learning is suitable and proved potential in regular traffic scenarios learning struggles with vast amounts of state and expansive behavioral memory. Second, the per-synchronized signal is the current time of traffic layout through top time. In this, we use DRL for signal optimization and manage a large set of states and tasks. A sign of our approach is that the signal should not be pre synchronize or rooted; rather, they use an active signal depending on simulating traffic. Another type is the Multi-agent Reinforcement learning approach which is used for signal or signal management involving multiple signal interceptions. Each interception is controlled by using independent reinforcement learning. As in Single-Agent Reinforcement learning.

3. BACKGROUND

In this part, we are going to analyse the shortcomings of using traditional reinforcement learning techniques and why we are using instead Q-learning and deep reinforcement learning when the number of actions and number of states is remarkably high.

3.1 Reinforcement Learning

It is a type of ML technique deployed by the algorithms whereby the ML model learns through the experience without having any prior information. In this technique, an RL agent works with the environment and makes appropriate changes to optimise the performance. It is to learn the optimal behaviour within an environment to get the maximum reward until the optimum performance is achieved. The RL agent must do some exchange in between catching arbitrarily activity for the surrounding and taking some optimal actions to exploit to get Q values, which is the looking forward reward of a pair that gets stored in a matrix depending on the information gained by the agent.

Q-learning is also known as temporal difference dominance algo, firm on the step-assess fun (X, A) air of state action. Observing the table of q, holds and updates q-values (positive and negative rewards) for every action at any state whenever an action is performed. Q-values determine how well the Action at a particular State using Bellman Equation. Managing the Q-table matrix is relatively easy for a lesser number of state-actions however, it becomes highly complex for fewer state-action spaces. Suppose if we solve a problem of more than 1000 sets of state-action space and each having N dimensions by Q-learning then we will have to maintain a q-table with a dimension of 1000 X no of actions and will be very hard to direct.

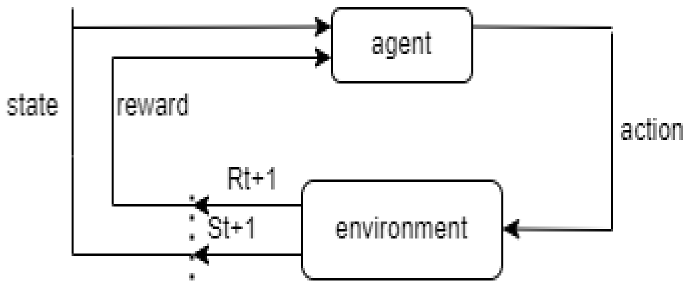


Figure.1 Reinforcement learning cycle

3.2 Deep Reinforcement Learning

It combines the RL framework with deep neural networks to solve high-dimensional complex tasks. As we have discussed previously, implementing real-world problems with classic RL and Q-learning techniques in tabular form is not appropriate since it becomes significantly complex to manage because of the high dimensions of a lot of new, unknown states. Reinforcement learning with deep neural networks can solve such high-dimensional complex real-world problems by combining function approximation and target optimization to help software agents reach their goals.

This paper used the DRL technique, uniting deep neural networks with traditional RL for such high-dimensional space problems.

4. TRAFFIC FLOW MANAGEMENT

This section describes the approach we took to ensure a balanced traffic stream across the network. To ensure our traffic control system can solve the real-world problem we operated to present traffic data to alter traffic lights accordingly using the existing Deep Reinforcement Learning Technique integrating the Q learning algorithm.

We trained our DRL agent based on collected information using the DRL approach to turn on the traffic signal more efficiently to solve traffic congestion problems by reducing the overall waiting time of the vehicle. We took it one step further using a rerouting approach to reduce the congestion for seamless traffic flow.

4.1 Traffic Light Optimization

For efficient traffic light control, we used Deep Reinforcement Learning. RL technique is based on the Q-learning and Actor-critic algorithm where positive or negative attributes are given to each component to optimize the working of the system by choosing pre-eminent steps for each declaration & estimating values of state function V(x). A detailed explanation of the reward-giving approach to state-action space is given below.

4.1.1 States

We represented the spot and speed of the vehicle as stated in our working model. In the environment, there is given t which is the agent step and it's denoted by (st) basically the state has to offer sufficient statistics approximately the distribution of vehicles on every road to permit the agent to productively discover ways to optimize the traffic.

4.1.2 Action

In our model, actions represent the turning and changing of traffic lights based on the congestion of traffic. To optimize the traffic light signal on the road we made shared based traffic lights in which the vehicles on opposite sides, which are going to turn to the right side from their respective spot share the same light, and the vehicles which decided to spin in left direction from own spot goes with same identical light. In our project we have taken time for green is 10 seconds and time for yellow is 4 seconds.

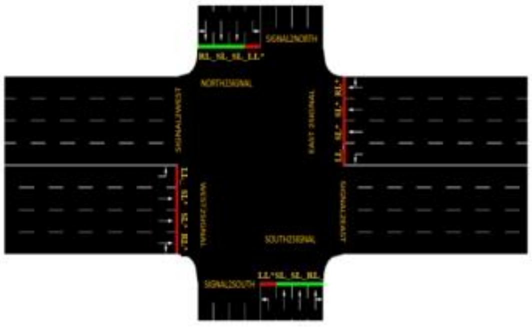


Figure.2 traffic light signals on road intersection

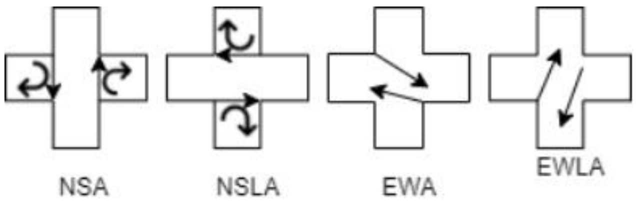


Figure.3 Four Possible actions

NSA: north south advance
NSLA: north south left advance
EWA: east west advance
EWLA: east west left advance

4.1.3 Reward system

The prize is given based on the overall time of awaiting of all vehicles in a single lane. The halt is recorded the moment an automobile takes entry into any single lane after that we total the halt time of the automobile on one road. Whether any of the vehicles leave their waiting time doesn't add up to the sum waiting time. On the basis of this reward, we increase the efficiency of the system by reducing total wait time. For our DRL agent, we use the following reward function.

Rewards = Old waiting time – New waiting time

Using this formula, if the present time for awaiting increases and is greater than the last time for awaiting the reward becomes -ve and then the agent gradually overcomes this circumstance on its own by reducing the new waiting time. Hence the Agent will send suitable signals to the traffic light controller to get maximum rewards.

4.1.4 Training and Neural Network Models

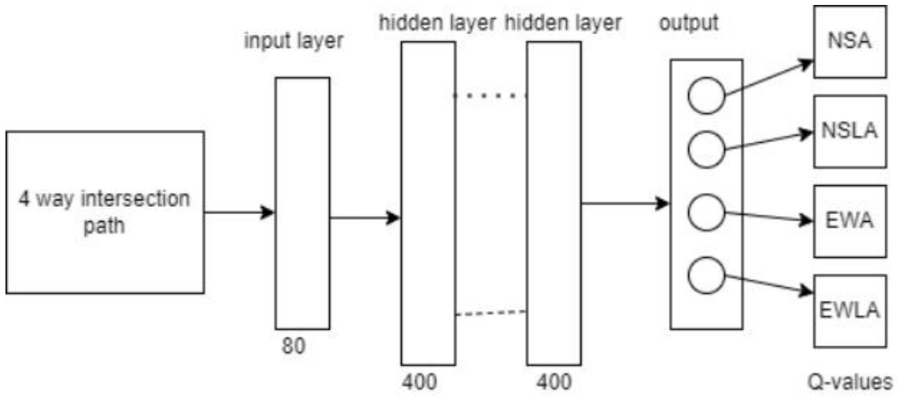


Figure.4 Deep neural network

We trained our DRL model by using neural networks and Q-learning together. That existing method is called Deep Q-networking. We are using three neural network layers for estimating the values of Q.We used three such models, everyone having an I/P layer and eighty I/P neurons and having a hidden layer with three-hundred, four-hundred, and one-hundred neurons, and an O/P layer with 4-neurons.

In our method, we used a Q-learning approach based on the value, our model picks out an efficient way for the entire three action and state sets, and the q value is generated on the basis of that action. The previous state-action pair output value is fed as input for the next state-action pair, and after some hidden layer processing, it gives new Q-values as an output.

The formula we used is as follows

Q (new-state, new-action) = Q (old-state, old-action) + β (rewards + d Q (new- state, new-action)) – Q (old-state, old-action)

Where d is the discount function (0, 1).

After training the neural network on the above formula, our trained network is ready to predict in real-time the best traffic action. When Vehicles wait for getting traffic signals in queues our trained model turns on the proper traffic light signal aiming to get maximum rewards given as follows.

Q (new state, new action) ← r (new state, new action) + d max_ a Q (new state, new action)

Where max_ a Q (new state, new action) it keeps all the Q values of the O/P of the neural network.

The best value is taken in respect of that action.

Best action = argmax(NNpredictedQ - values).

A = argmax_a Q(current-state, current-action).

During the period of training, the agent of our DRL model will find the present state of the road, the actions of an automobile, and the prize of that action. Each data gets loaded in the buffer memory of replay and used in training our model. At the start of training our agent takes some action value on random and some from the Q-network on the basis of the epsilon greedy policy. The difference of square of the real and predicted values is the loss function we used in the model, and then the loss value is minimized by adjusting and updating the weights.

Loss = (reward + d max_ a Q(current state,current action; θn) – Q(previous state, previous action; θo)) ^2

During training, the calculation of the actual value is done by the formula = R + d max _a Q (old state, old action). For every N number of steps, we assign the actual weight to the target network.We took one batch of the sample data and fed it to the I/P layer with a vector variable containing eighty input values since the input layer got eighty neurons. Whereas the output layer with four neurons gives four Q-values each representing the possible actions. We used the RelU activation function to output the result. We used Mean Squared Error to analyse the network output with the desired result.We used the Adam algorithm for the optimization of the loss function.

4.1.5 Epsilon Greedy Policy

At the beginning of the training since all the values are zero, we use the Epsilon Greedy approach to fill the value. In this approach, our agent freely explores the environment without any fear of getting high negative rewards. This free exploration helps the agent to acquire familiarity with new states. Exploration is done based on a given epsilon value, EGP creates a no. between zero and one and if the random values are higher than ε value the agent's exploration gradually decays, and it takes better actions to exploit the environment based on prior knowledge of the agent. That means our agent becomes greedy and starts exploiting the environment more than exploring the environment to reach max reward value.

The epsilon value decays with each episode as follows:

Epsilon value = 1-Current Episodes/Total Episodes

4.1.6 Agent's Memory

Our DRL agent works on the prior experience and knowledge to take action which gives maximum rewards. Q-values determine how good the Action is at a particular State using Bellman Equation. We created a stack to store the previously gained knowledge of our agent. The stack stores the sample of each episode, which comprises the current state, old states, old actions, and rewards. Once the stack is full the oldest rack of samples gets deleted and the most recent sample gets added.

4.1.7 Experience Replay method

This method works on weight updation in neurons, choosing a random batch of samples from the memory stack in the training phase. To optimize the training of the agent, if we choose two consecutive samples from the memory stack there is a higher chance of correlation which can affect the accuracy of our model. We take two hundred sample groups picked at random in the training process to solve this.Our agent is testing on one hundred episodes, each episode having two thousand and five hundred simulation steps.Our well-trained agent uses the current state of the road intersection to calculate waiting time and length of the queue for each section of the road.Since the reward is calculated based on the formula, Rewards = Old waiting time – New waiting time,our agent takes necessary actions with the help of the replay memory to maximize the reward. It turns the traffic signals to yellow if the old action and new action are not equal and if both are equal the traffic signal turns green. This means our model is working on optimizing the traffic more efficiently.

5.EXPERIMENTAL SETUP

In experimental we have done experiment in synthetic environment SUMO (simulation of urban Mobility), basically it is a traffic simulator in this we can we do simulation of traffic in real time, it's manages the delay timing of traffic.

5.1 Properties of Network

An in depth description of road community homes, site visitors policy, and sensors is given with different essential information in this subsection.

5.1.1 Lanes and Road

The tracery of roads is used for simulation. There is more than one way to attain one position to any other in the tracery. Some lanes have traffic lights, and some lanes have traffic lights paths but no signal, and we know that the paths without signals are longer. We selected the total length of the road: from all four directions to the intersection,800 meters (750+50), the length of diagonal we can find by using Pythagoras theorem is 1131 approximately we say 1140 meters.

Paths without signals are available diagonally across the road. Roads are traffic signals. The two central lanes on either: vehicle going direct. And remaining vehicles don't follow a direct path they go left or right.

Attributes	Value
Environment for simulation	Simulation of urban mobility (SUMO)
no of the vehicle per episode	980
no of traffic light	1
Time for simulations	5400 second
Time of simulation for all vehicles	98 min
Intersection type	4 ways

5.1.2 Route

2 different routes; the 1st one is fixed which get setted prior the simulation, second one is the dynamic route which is get applied with the support of Traci fixed for traffic situation, and conditions while performing on the road. If a static route is assigned, 60% of vehicles can travel on the direct route and only 40% of vehicles are allowed to turn left or right.

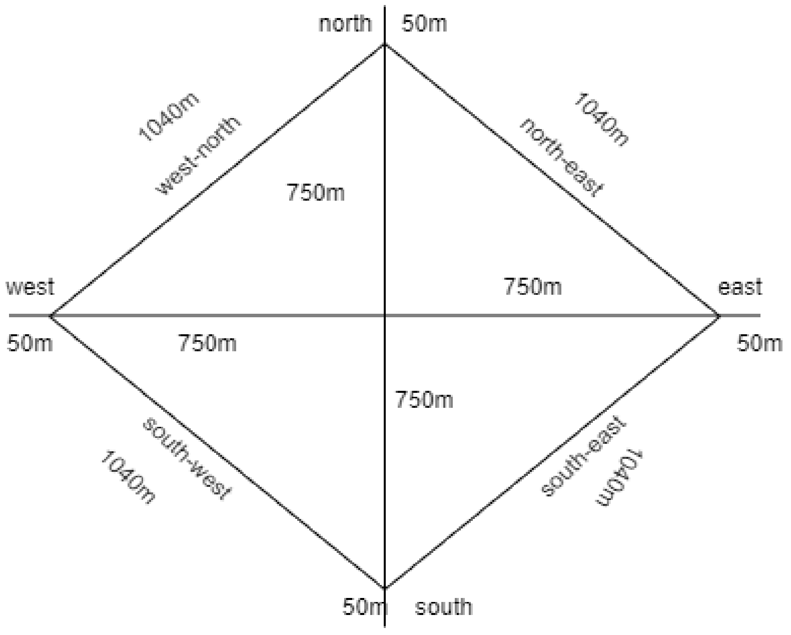


FIGURE 5. Road network configuration

5.1.3 Traffic Generation

In this section, we took

1000 vehicles for testing different models of neural networks. A new car joins alternative sec. Whenever another latest vehicle attaches the tracery, it is assigned a way according to stopping place. We use automobiles for simulation having unlike velocities. All automobile follows a path relative to their station, conversing the small road. setup simulation 60% of the traffic is facing each other along the shortest route from origin to destination, although each vehicle must pass through a traffic light intersection. The remaining 40% of vehicles ought not to move between regulated intersections. The automobile might choose a substitute route, a route without a signal, which will in some cases be the fastest route with respect to the calculation of timing.

5.1.4 Weibull Density Function of Probability

Automobiles at a crossing could determine by simulating the entry of vehicles on the crossing at a particular time span. This can be concluded by enumerating the time span joining 2 consecutive automobiles coming at the crossing. We have used the following distribution function of probability.

f(T) = βηβη\frac{\beta}{\eta} (T−γη)β−1(T−γη)β−1(\frac{T-\gamma}{\eta})^{\beta-1} e(T−γη)β

Training Stipulations and values of the agent

Stipulations	value
Episodes	100
Graphical User Interface	false
Gamma value	0.70
Size of the batch	98
Size of memory	50000
states	80
action	4
Max steps	5400
duration for green	10
duration for yellow	4

β *is slope,* η *is scale,* γ *is location*, You only need a scale parameter for vehicle arrival time, then by if we set γ is equal to 0 and let's suppose β=C=constant and distribution became parameter.

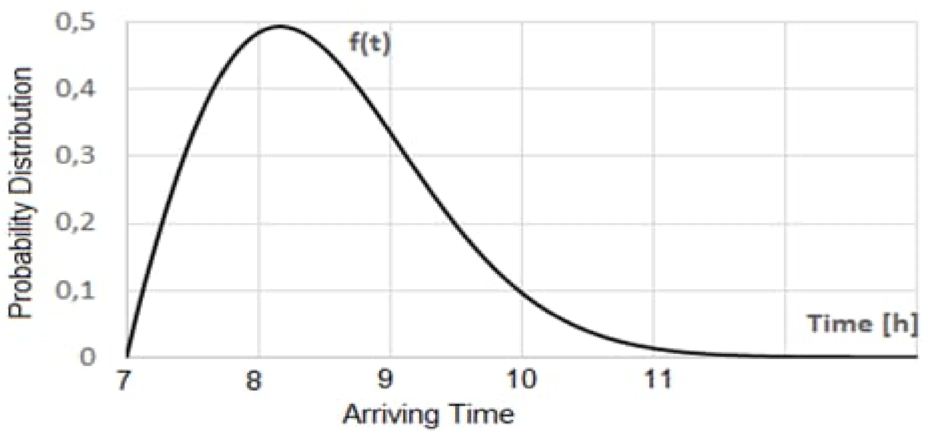


Figure 6. Weibull Distribution for vehicle arrival.

5.1.5 Performance parameter

There are some performance parameter for evaluating performance like reward, Delay, queue length, simulation time.

5.1.5.1 Reward

For the reward maximization, the agent should take action, the normal tendency of all networks is to attain negative rewards which approach 0, which means the agent learns slowly.

5.1.5.2 Delay

In the traffic when vehicles get stuck, they wait till the traffic is clear, hence delay time increases.When agents know about the environment the cumulative delay decreases gradually.

5.1.5.3 Queue Length

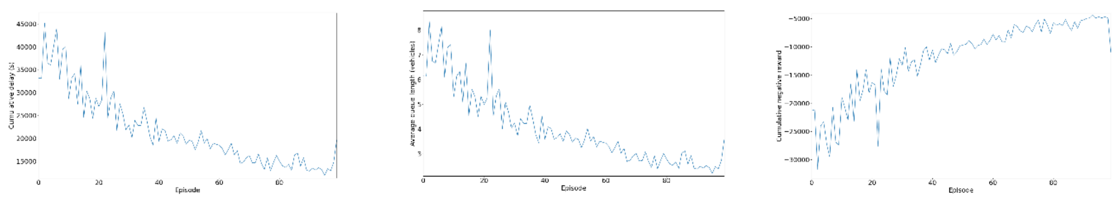
If the road is congested and the traffic lights are shut, there will be a line of automobiles holding on for the traffic lights (green) to turn on. We have selected the average queue length of parameter of performance of agent. It's inversely proportional to episodes.

5.1.6 Time for Simulation

Simulation during the final automobile way out of the simulation. The max time is 5400 seconds which is 1:30 hour if we lessen the rush by the RL agent, in that case, the vehicle will wait less and it will take less time to reach the target.

6. RESULT AND DISCUSSIONS

For the initial experiments, in this, we have used 300 neurons for training the model for determining all three parameters and observe the graph we got and compare it with two other models to determine the performance and analyzing the graph for comparing the result we used deep reinforcement learning. We used the DRL model for the creation of traffic lights for making them more functional by observing the output. In the last aggregate model in which we change the value of gamma, in the CNR we can see two lines are there one is for 0.4 and the other is for 0.70. In that we can see that for the value of 0.4 is giving a better result, we want to make the q value more efficient. If we increase the traffic then we can see that the value 0.4 gives not a good and stable curve. That all happened because of high traffic, if we increase the vehicle thus increasing traffic it leads to more time for waiting for the vehicle.

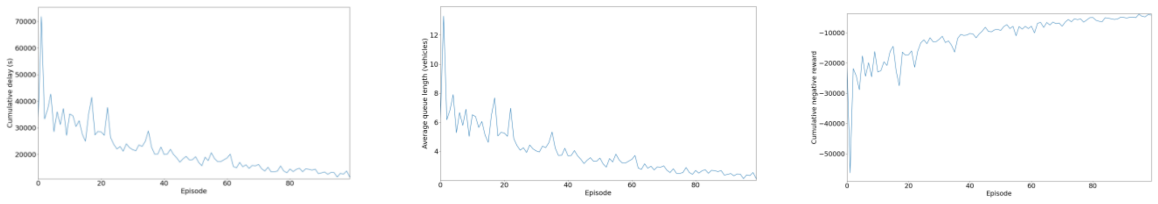


CD AQL CNR

Fig.7 model 1 for 300 neurons

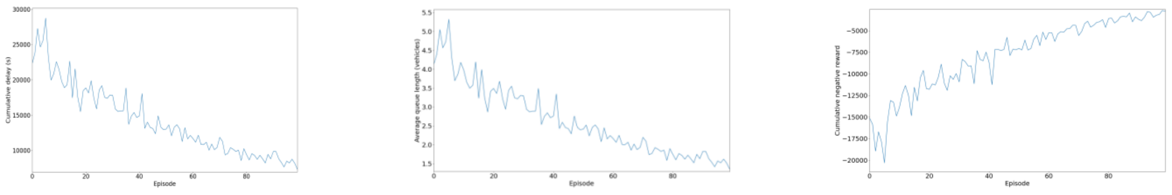
6.1 Traffic Light Implementation

In this section, we are controlling the traffic, by the help of reinforcement learning we have created a traffic light controller, in this traffic light we have calculated the situation of the intersection. In this, we have calculated all the possible parameters like the delay of vehicle, and length of the queue. The work of the agent is to reduce the delay time, AQL, and NR. We trained the agent on three different models with different neural network architectures, for model number 1 we have taken 300, for model 2 we have taken 400, for model three 100 neuron layers.



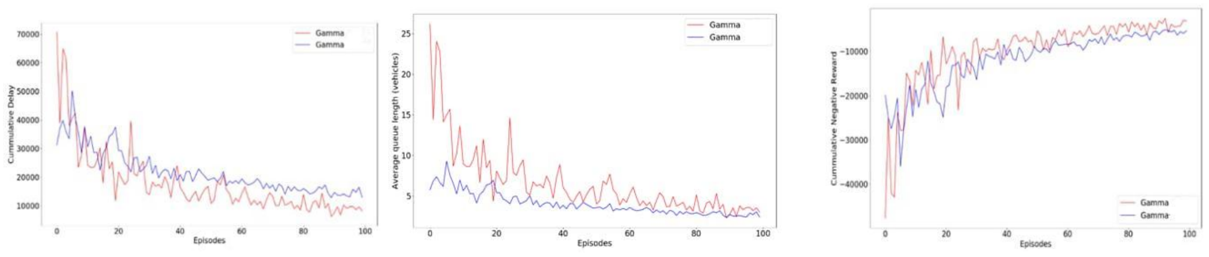
CD AQL CNR

Figure.8 model two with 400



CD AQL CNR

Figure.9 model with 100 neurons



CD AQL CNR

Figure.10 change values, 0.40, 0.70

By observing the graph, we can see the performance of the neural network is inversely proportional to the hidden layer.

100 layers neural network gives a very good performance, we can observe a little decrease in waiting time due to the decrease in congestion which is achieved by efficient traffic light operation by our DRL agent.

6.2 Changing Gamma Value

In the training setting in agent, we have taken different values of gamma. If we will take the value of gamma 0.40 then it will show a blue-color line, if we take the value of gamma 0.70 then it will show a red colour line. By comparing both values we observe there is a small variation in that graph and for gamma value 0.40 shows a better outcome as compared to 0.70.

7. CONCLUSIONS AND FUTURE DEVELOPMENT

In this project, we successfully implemented reinforcement learning for solving problems for traffic flow control and its management, in this project we simulated this model in a SUMO environment which provided simulation-based testing for our reinforcement model and it also helped in evaluating our reinforcement model visually and gave us an overall overview of that model that we have trained in a realistic figure like view of vehicles. We can see the vehicle running in all defined directions we can set the delay time according to convenience and we can control the flow control of traffic in the SUMO environment. Future of this project can be, as we can see day by day number of vehicles is increasing on the road, and in a developing country like India traffic is one of the most poorly managed systems, if we implement reinforcement learning in traffic light control for a four-way intersection of the road than we control the traffic more efficiently, and if we can implement it on four-way intersection then we can possibly do the same with 6-way, 8-way, 10-way intersection for better handling of exponentially increasing traffic, it can give better solution for the huge traffic problem.

REFERANCE

[1] R. S. Sutton, A. G. Barto et al., Introduction to reinforcement learning. MIT press Cambridge, 1998, vol. 135.

[2] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke et al., “Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation,” arXiv preprint arXiv: 1806.10293, 2018.

[3] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, “A survey on reinforcement learning models and algorithms for traffic signal control,” ACM Computing Surveys (CSUR), vol. 50, no. 3, p. 34, 2017.

[4] W. Genders and S. Razavi, “Evaluating reinforcement learning state representations for adaptive traffic signal control,” Procedia computer science, vol. 130, pp. 26–33, 2018.

[5] W. Genders and S. Razavi, “Using a deep reinforcement learning agent for traffic signal control,” arXiv preprint arXiv:1611.01142, 2016.

[6]P. Koonce and L. Rodegerdts, “Traffific signal timing manual.” United States. Federal Highway Administration, Tech. Rep., 2008.

[7] L.-J. LIN, “Reinforcement learning for robots using neural networks,” *Ph.D. thesis, Carnegie Mellon University*, 1993.

[8] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, “Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network,” arXiv preprint arXiv:1705.02755, 2017.

[9] S. S. Mousavi, M. Schukat, and E. Howley, “Traffic light control using deep policy-gradient and value-function-based reinforcement learning,” IET Intelligent Transport Systems, vol. 11, no. 7, pp. 417–423, 2017.

[10] R. Dowling, “Traffic analysis toolbox volume vi: Definition, interpretation, and calculation of traffic analysis tools measures of effectiveness,” Tech. Rep., 2007

[11] C. J. Watkins and P. Dayan, “Q-learning,” Machine learning, vol. 8, no. 3-4, pp. 279–292, 1992.

[12] C. J. C. H. Watkins, “Learning from delayed rewards,” Ph.D. dissertation, King’s College, Cambridge, 1989.

[13] J. N. Tsitsiklis and B. Van Roy, “Analysis of temporal- difference learning with function approximation,” in Advances in neural information processing systems, 1997, pp. 1075–1081.

[14] L.-J. LIN, “Reinforcement learning for robots using neural networks,” Ph.D. thesis, Carnegie Mellon University, 1993.

[15] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, T. Ewalds, D. Horgan, M. Kroiss, I. Danihelka, J. Agapiou, J. Oh, V. Dalibard, D. Choi, L. Sifre, Y. Sulsky, S. Vezhnevets, J. Molloy, T. Cai, D. Budden, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, T. Pohlen, Y. Wu, D. Yogatama, J. Cohen, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, C. Apps, K. Kavukcuoglu, D. Hassabis, and D. Silver, “AlphaStar: Mastering the Real-Time Strategy Game StarCraft II,” https://deepmind.com/blog/ alphastar-mastering-real-time-strategy-game-starcraft-ii/, 2019.