IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 3, 2021, accepted February 15, 2021, date of publication March 2, 2021, date of current version April 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3063463

Traffic Flow Management of Autonomous Vehicles Using Deep Reinforcement Learning and Smart Rerouting

ANUM MUSHTAQ[1], IRFAN UL HAQ[1], MUHAMMAD USMAN IMTIAZ[1], ASIFULLAH KHAN[1,2], AND OMAIR SHAFIQ[3]

[1]Department of Computer and Information Sciences, Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad 44000, Pakistan
[2]PIEAS Artificial Intelligence Center (PAIC), Pakistan Institute of Engineering and Applied Sciences, Islamabad 44000, Pakistan
[3]School of Information Technology, Carleton University, Ottawa, ON K1S 5B6, Canada

Corresponding author: Anum Mushtaq (anumkhan.26@gmail.com)

ABSTRACT Autonomous Vehicles (AVs) promise to disrupt the traditional systems of transportation. An autonomous driving environment requires an uninterrupted, continuous stream of data and information based on complex traffic data sets and predictive measurements to make critical and real-time decisions in uncertain situations. Such an environment fosters a self-organizing system where vehicles must be seamlessly connected and various other services and intelligent decisions to manage traffic flow must be executed in an emergent manner. To proceed towards this vision, in this paper, we develop a traffic flow management model which is based on a novel two-phase approach for AVs to optimize traffic during congestion periods. In the first phase of our approach, we build an adaptive traffic signal control, using Deep Reinforcement Learning (DRL) to optimize traffic flow on road intersections during the periods when traffic is congested. In the second phase, we implement a Smart Re-routing (SR) technique for the traffic approaching intersections. Re-routing is used to carry out load-balancing of traffic to alternate paths to avoid congested road intersections. The experimental evaluation of the proposed approach is validated using simulations that demonstrate up to 31% improved performance efficiency compared to the traditional settings using pre-timed signals and without re-routing. The two-phase approach improves the overall traffic flow while reducing delays and minimizing long traffic queues' lengths. This approach is useful for making infrastructure intelligent enough to handle traffic congestion and balance traffic flow efficiently.

INDEX TERMS Traffic management, reinforcement learning based traffic control, autonomous vehicles, routing of vehicles, traffic control.

I. INTRODUCTION

Autonomous vehicles have opened new avenues in the area of Intelligent Transportation System (ITS) [1]. By incorporating intelligent control techniques, ITS has an immense potential to revolutionize the coordination between vehicles and road infrastructure [2]. This gives rise to a novel system of mobility called Automated Highway Systems (AHS) [3], which incorporates intelligence at various levels of a transportation network.

Autonomous Traffic Management (ATM) is one of the most vigorously pursued areas of research in the ITS now a days [4], [5] which promise to reduce traffic problems through a well connected and coordinated infrastructure. There are many challenging issues in the present traffic systems, one of which is traffic congestion [6] that requires seamless management while maintaining traffic safety [7]. Considerable progress is required to make autonomous vehicles behave smoothly and reliably in heavy traffic during loaded hours. This needs a mechanism to learn traffic patterns in real-time and perform predictable measures to optimize traffic flows and minimize congestion. For this, AVs and their connected environment need to manage through dynamically changing traffic situations intelligently, and road conditions [8], [9].

Numerous vehicles nowadays depend on intelligent systems for many decisions such as Lane-Keeping Assistance (LKA) and Collision Warning

The associate editor coordinating the review of this manuscript and approving it for publication was Chih-Yu Hsu.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

Systems (CWS) [10] etc. These systems are now mature enough to guide autonomous vehicles in critical decision-making to avoid accidents. To alleviate congestion issues, traffic management systems may utilize infrastructures such as traffic signs, traffic lights, roadside units, and other similar controls [11]. However, when there is an increasing number of vehicles approaching a specific area, traffic congestion also proliferates. The challenges are to (1) coordinate autonomous vehicles in-advance, (2) properly manage different traffic situations, and (3) improve traffic flow rather than only avoiding congestion. To address these challenge, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) coordination is essential [8], [9]. Another critical challenge in managing traffic flow, especially in autonomous vehicles, is the management and scheduling of intersection networks. Traditional deterministic models for the management of traffic do not scale well for large networks in urban settings [12]. When traffic is heavy at intersections during peak hours, congestion affects not only upstream traffic but at all intersections. Thus, there is a need for an efficient method that should schedule complex and dynamic traffic situations. Autonomous vehicles need more intelligent strategies and intelligent infrastructure to cope well in complex and unstructured environments.

Reinforcement Learning (RL) is a technique used in an artificial intelligence's paradigm to teach autonomous vehicles how to learn from their environment by some reward and punishment mechanism [13], [14]. The relevance of RL becomes significant for autonomous driving because of its ability to intense interactions with other vehicles, road networks, pedestrians, and the environment, which is difficult to pose as a supervised learning problem [15]. Neural networks and RL provides a sophisticated framework for managing and maintaining traffic conditions [16], [17]. RL can make autonomous vehicles or infrastructure intelligent through learning by mistakes and interaction with the environment. Sensors technology combined with RL can make autonomous vehicles informed about the upcoming obstacles and help make proactive decisions to avoid these obstacles.

AVs constitute multiple tasks and stochastic traffic scenarios. This research focuses on automating the traffic light system by using DRL technique and dynamically rerouting the trailing traffic. Existing infrastructure for traffic light control is inefficient and cannot cope well with autonomous vehicle systems, and has many associated problems, such as energy waste and long delays [14]. Moreover, there may be conventional transitions of traffic lights. For example, on one side of an intersection, we have several cars stuck in a traffic jam, and on the other side, there may not be any car, but the signal is open for that side due to the pre-timed fixed signals. Vehicles halted in a traffic jam have to wait for a predefined time for the traffic light to turn green. The uncertainty and continuously changing dynamics in the environment would make stochastic cost function maximized and predict optimal solutions at each instant and learn new configurations of the driving environment.

To address these challenges, the contribution of this research is to manage traffic by a two-phase approach:

- In the 1st phase, we have implemented a DRL based mechanism to build an adaptive traffic signal control to reduce traffic congestion on the intersection. We train our learning agent to optimize traffic signals such that it turns on the appropriate traffic light so that the overall cumulative waiting time of all vehicles reduces.
- In the 2nd phase, we are monitoring the congestion situation periodically for traffic before it approaches a traffic signal intersection. When it exceeds a particular limit, we implement a smart routing module to reroute vehicles to alternate paths by doing computations on the routes.

We have developed simulations in SUMO where we used DRL to adjust traffic signals dynamically depending upon each vehicle's long queue lengths and waiting times. We applied rerouting on preceding vehicles that have not yet reached the intersection. Due to congestion on intersections, we adjusted certain vehicles' limits that help reroute vehicles to alternate routes from route tables based on each vehicle's origin-destination matrix.

The rest of the paper's organization is as follows: Section II describes the field's literature review. Section III overviews the background in which essential concepts related to the RL and DRL are discussed. Section IV describes the proposed approach for traffic flow management. Section V contains the experimental setup for the proposed approach. In section VI, there are results and discussions, and section VII concludes the paper.

II. RELATED WORK

Machine learning and artificial intelligence techniques contribute to almost every field nowadays [18], [19]. Several techniques and strategies are available, ranging from traditional to futuristic to improve traffic flow and congestion management [20]–[23]. Congestion occurs when traffic exceeds road capacity, consequently travel times and queue lengths are increased, and speeds lower, especially during hours when traffic demand is high [20], [21]. We can divide the congestion problem solutions into two main aspects that primarily target traffic flow management. In the first aspect, traffic infrastructure plays a role in congestion management and coordination of traffic signals. There are many research contributions available for the improvement of signal timings over a few decades. For example, in [23] the fuzzy logic table measures the signal duration, in [24] authors proposed a neuro-fuzzy controller, and in [25] dynamic programming is used to solve this problem. The second aspect deals with the road infrastructure, which plays a vital role in congestion management. RL method has been used as a traffic control problem since the mid-1990s, but with the advancement in AVs, the interest in this technique has immensely increased [26]. Traffic control by DRL has great potential as compared to conventional approaches [27], [28]. There are many use-cases in this field showing promising performance [22], [29]. SARL (Single-Agent RL) is used to

control traffic where a single agent is responsible for control a single junction. In [30] SARL is used to manage an isolated junction, which is signalized as two-phase. Pre-timed signals under constant flows and variable flows have been used for Q-learning agents. Results were used to adapt to different conditions of traffic. Another study based on single-junction with three Q-learning agents is presented in [31]. They used different state representations, such as vehicle arrival to the green light and queue at the red light, cumulative delay, and queue lengths. The study is performed in heavy traffic and shows promising results. SARL approach is suitable for single signalized junction and proved its potential for conventional techniques. Q-learning in traffic scenarios suffers from the massive volume of states and actions, which expands the memory size. Secondly, pre-timed signals cannot tackle real-time traffic scenarios during peak hours. We use deep reinforcement learning to optimize the signals and to manage large state and actions sets. Our approach's signals are not pre-timed or fixed; instead, we used dynamic signals based on traffic situations.

Another type is the Multi-Agent Reinforcement Learning (MARL) based approach used for signal control contains multiple signalized intersections. An independent RL agent is used to control each intersection, just as in SARL. MARL approaches could be used for large networks. In [32] authors developed a reinforcement learning-based approach for adaptive signal control and present three algorithms TC1, TC2, and TC3. Results show that using fixed time controllers, RL outperformed on a simple 3:2 grid. Authors also implement co-learning where driver agents and signal controllers learn value functions. Optimal routes through the network can also be found by learning compute polices. This was the most basic work and had many limitations, such as increment of the state space and fixed time controllers. This work is extended by [33] by implementing necessary information sharing between RL agents. The authors introduced three new traffic controllers named TC-SBC, TC-GAC and TC-SBC+GAC. A congestion bit is added in TC-SBC, which is used to know traffic conditions at neighbouring intersections. TC-GAC uses this information for optimal action selection. Algorithms tested on simple grid networks performed well under different traffic flows. The limitation of TC-SBC is that it increases state space size, which is challenging to compute, and TC-GAC does not learn permanently about traffic flows or congestion. In [34] further enhancement of this work is done by introducing two new algorithms TC-SBA and TC-SBAC. Extra bits to state and accidents are added, which cause it to suffer again from the large state space. The new state space is exceeded four times the original TC1. Another implementation of MARL is done in [35] on large networks of almost 50 junctions. The algorithm is based on Q-learning, and for state representations, the average queue length at each link on every junction is considered. Fixed time signal plans are used, and a green time-ratio is used for action selection.

Due to high-dimensional state and action spaces while applying RL results in complex control problems. To solve this problem, a combination of RL with deep neural networks has shown promising results. The authors of the study performed in [36] used value function-based agents and deep policy-gradient to predict the optimal signal at intersections. A snapshot of the current state is received to these adaptive traffic light controls to produce signals at each time step. A large volume of graphical data is challenging to manage, delayed, or wrong data interpretation could mislead and cause severe impacts. Multi-agent system using RL for efficient signal control is studied in [37]. The study used two types of learning agents, i.e. an LQF based outbound agent for scheduling traffic signals and a central agent for learning value function from its locals and neighbours traffic conditions. LQF algorithm is a limited approach because waiting time in other intersections could increase if the number of vehicles is low. Time delay due to interruption of outbound agents also affects the case of the AVs. In [38] author developed a learning algorithm named RMART for controlling signals. In this approach, information sharing among neighbouring controllers is done by I2I communication, which improves signal controls' performance, but this could cause worse traffic situations if communication between controllers fails if used in an AV's environment with no driver to handle these kinds of problems. They modelled complex traffic scenarios and showed promising results. In [39] authors developed an intersection management system using a discrete consensus algorithm to coordinate agents. Excessive traffic is distributed among other intersections using a rerouting algorithm. The authors tried to balance traffic and reach out to an equilibrium state during the worst traffic situations. The authors used rerouting algorithms, but there is no proper traffic control mechanism on the intersection discussed. The AVs era requires traffic management solutions based not only on the combination of strategies that take into account a high-level view of traffic scenarios.

Early techniques have used deep reinforcement learning and rerouting for traffic congestion and management, but there are few limitations in the existing literature: (1)Most of the studies only consider signal control on intersection or rerouting problem, which is not very useful. (2)The signals are usually pre-timed fixed interval which could worsen traffic condition in many situations. (3)Pre-timed fixed-route plannings based on origin-destinations can make AVs stuck in severe traffic congestion and hours of delays to reach their destinations. To address these limitations, in this paper, we used a combination of techniques for our approach that will intelligently manage traffic signals using deep reinforcement learning while smartly rerouting the traffic behind the intersection to avoid more congestion. We used dynamic signal control that is using real-time traffic information to turn on and off. We are also using real-time traffic data to rerouting the vehicles to alternate routes. Traffic signal suffers from high dimensions of states and action as there are many state-changing in traffic scenarios. DRL-based signal controls solve high dimensions of conditions efficiently.

III. BACKGROUND

This section will review why we are using deep reinforcement learning, and Q-learning instead of traditional reinforcement learning. What are the limitations of classical RL techniques when the number of states and actions becomes exceptionally high?

A. REINFORCEMENT LEARNING

Reinforcement learning is based on algorithms that can learn through the experience without any previous information or training. In reinforcement learning, an agent employed in an unknown environment interacts with that environment and takes suitable actions to maximize its performance [34]. A scalar reward (positive or negative) is received based on selected actions, and the agent continues to learn until the best performance is achieved. Agent in an RL environment must do some trade-off between exploration (taking random actions to explore the environment to maximize reward) and exploitation (agent only takes known optimal actions). Q values stored in a matrix are the expected reward of a state-action pair based on knowledge learned by RL agent [40].

In Q-learning system selects an optimal control-algorithm based on an action-value function Q(x,a) for all state-action pairs [41].A q-table is maintained for every action at any state. Positive or negative rewards can be added in the q-table, and it is updated whenever a new action is performed. Maintaining a q-table for low dimension state-space is possible, but it becomes very complex when the number of states and action space is very high. Suppose state is a high dimensional vector of N length where N is a big number. If we solve this problem by q-learning and set state space to 100 vectors each of N dimensions. Q-learning needs a table of dimensions 100 x number of actions which becomes very difficult to manage.

B. DEEP REINFORCEMENT LEARNING

DRL can solve complex high-dimensional tasks and could apply to dynamic data [42]. Implementing real-world problems in the tabular representation of state and action space becomes very complex to imply because of the significant number of new and unknown states. In AVs' traffic signal optimization problem, such a large volume of data becomes very complex to represent in Q learning tables due to high dimensions. RL with some function approximators [43], known as DRL, can be used to solve this problem. Function approximators can be of two types: linear function approximation, in which a linear combination of states, actions, and learned weights are used and non-linear function approximation like a neural network, we can use deep neural networks. This paper used a non-linear function approximator, a deep neural network, to deal with such high dimensional space under RL's classical setting.

IV. TWO-PHASE APPROACH FOR TRAFFIC FLOW MANAGEMENT

This section will give a detailed description of our approach to make a balanced flow of traffic through the network. Our approach is based on the existing Deep Reinforcement Learning (DRL), and specifically Deep Q-Learning approaches. To make an efficient traffic control system, we use real-time traffic information to adjust traffic signals dynamically. We utilize a DRL approach to select the best traffic signals to turn on, based on collected data from infrastructure. To reduce congestion, we train our learning agent to optimize the traffic signals to turn on the appropriate traffic light—the overall cumulative waiting time of all vehicles decreases. We further empower this approach by using the rerouting technique to reduce congestion and increase traffic throughput. Fig.1 shows an overview of our strategy for traffic flow management and optimization.

A. TRAFFIC LIGHT OPTIMIZATION

We used DRL approach for traffic light optimization. The system is guided by a reinforcement learning module and is indirectly controlled by an evaluation signal regarding actions and performance. Reinforcement learning is based on credit assignments where reward or punishment attributes are assigned to individual elements to maximize performance. Two main types of algorithms can control problems in reinforcement learning: Q-learning and actor-critic learning. Actor-critic learning can be further divided into two parts: One chooses optimal action for each state, and the other estimates state value function V(x). We will now explain in detail what states, actions, and rewards are used in our case.

1) STATES

In our model, states are represented as the speed and position of the vehicles. The intersection's road network is divided into small sections, and states are obtained from the sensors embedded on each section, which returns one if there is a vehicle present and returns zero if there is no vehicle. In each section, there could be only one vehicle at a time to reduce the computation cost, so the position dimension is a binary value. The speed of the vehicle is in m/s. When the velocity of the vehicle becomes zero, it means that the car got stuck in congestion.

2) ACTIONS

The operation performed by traffic light based on the current traffic situation is represented as actions. The traffic light should be optimized to take appropriate actions according to the problem. Usually, the vehicles from opposite sides want to move straight or turn right share the same traffic signal. On the other hand, vehicles on opposite sides that want to turn left from their respective ends share the same signal. By sharing the same signal means both of the traffic lights will turn green at the same time. Traffic lights are connected and turn on and off at the same time are shown in following table.

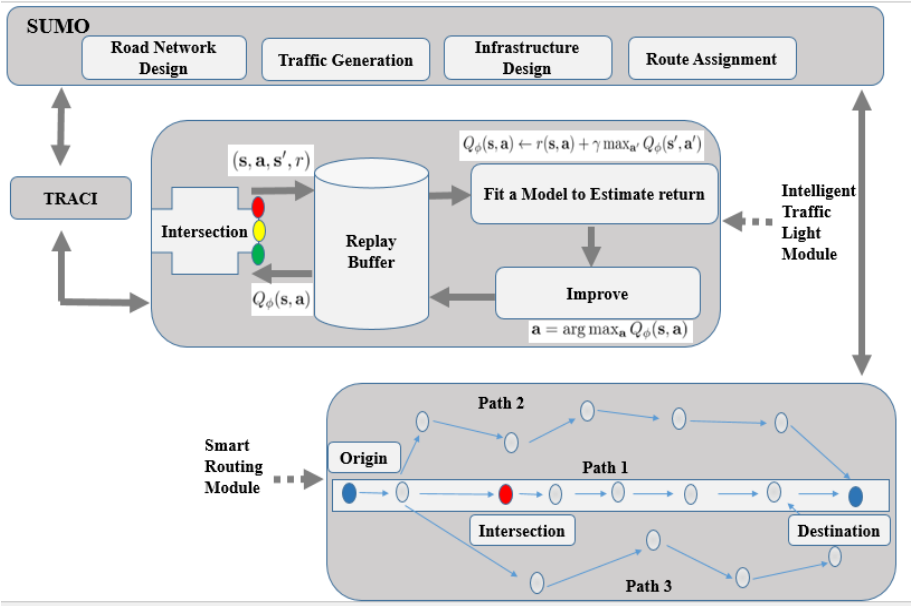


FIGURE 1. Two-phase approach for traffic flow management: Ist phase: Intelligent traffic light module. 2nd Phase: Smart rerouting of vehicles.

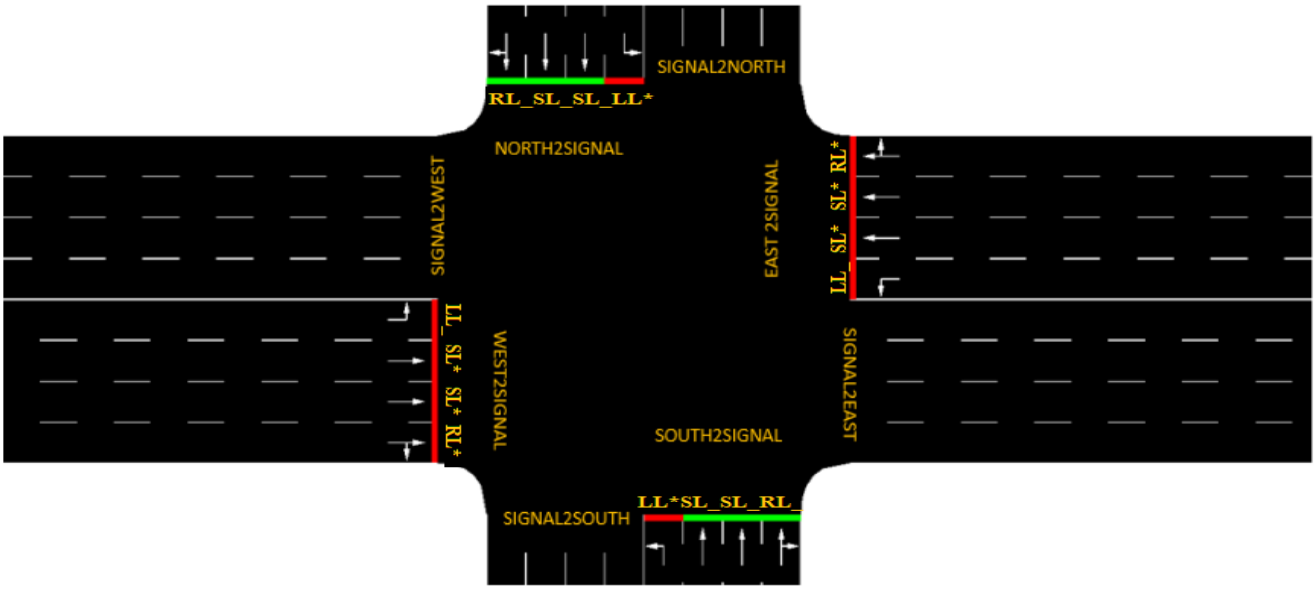


FIGURE 2. Operational settings for traffic light signals on road intersection.

Here LL means Left Lane turn, RL is Right Lane turn, and SL is straight Lane. Fig.2 shows an overview of this operational setting for traffic light signals.

Connected Opposite Directions	
LL* LL_ SL_ SL_ RL_ SL* SL* RL*	LL* LL_ RL_ SL_ SL_ SL* SL* RL*

3) REWARDS

A reward is the most important aspect of reinforcement learning, as it indicates a specific action's performance. Based on the reward next action is decided, so the reward should be clearly defined to get the best action. For our agent, the reward is based on the "cumulative waiting time". We record the waiting time when a vehicle enters a specific lane. Then, we add waiting times of all vehicles in that lane, but if the vehicle leaves that lane, its recorded time will not be added up in cumulative waiting time. As our main objective is to increase efficiency, the reward is based on reducing total

waiting time. We used the following reward function for our agent:

$$R = O_{wt} - N_{wt} \tag{1}$$

where R is Reward, O_{wt} is the old waiting time and N_{wt} is the new waiting time. It is clear that if the current state's waiting time is more than the previous one, it means the reward is negative, and the agent tries to overcome this situation gradually. It will take action and set the appropriate traffic light on the signal intersection that maximizes its reward.

4) TRAINING OF NEURAL NETWORK MODELS

The training is done by using an existing approached known as Deep Q-Network (DQN), approximating the Q values using three layers. The input layer accepts information such as data coming from sensors in our case. The hidden layers process the data, and output layers produce the output. Some weights are learned and updated at each layer. We implemented three neural network models; each has an input layer with 80 input neurons and an output layer with four neurons. The number of neurons in the hidden layers of first, second and third neural network is 400, 500 and 200 respectively. In Fig.4, the structure we used for our neural network model is given.

In our approach, we used value-based Q-learning in which the system selects an optimal algorithm for all state-actions pairs based on an action and Q values against that action, represented as Q(s, a). The input is given to the neural network as previous state-action pairs and after doing certain processing it outputs the new Q-values. Given below are the equations and algorithms which are based on and utilize Deep Q-Network (DQN).

$$Q(s_n, a_n) = Q(s_o, a_o) + \beta\,(r + \gamma Q(s_n, a_n)) - Q(s_o, a_o) \tag{2}$$

Heres_o denotes the old state, a_o is old action taken, r represents the reward given on taking the action, discount factor (0,1) is represented by γ, s_n and a_n is new state and new action.

A neural network learns the parameters by training, and final network is ready after the training process. Trained network is used at the prediction time to predict the best action in the current situation. During traffic light optimization, at the traffic intersection, when AVs are waiting in the queues to turn on the proper light phase, our agent takes the best action at state$s$ using the trained neural network, in order to maximize reward.

$$Q(s_n,\ a_n) \leftarrow r(s_n,\ a_n) + \gamma\ max_{a'}\ Q(s_n\, a_n) \tag{3}$$

The $\max_{a'}\ Q(s_n\, a_n)$ is the list of Q values contains maximum of neural network's output. We take the best values corresponding to our action. The $best\ action\ = argmax(NNpredictedQ - values)$.

$$a = argmax_a Q(s,\ a) \tag{4}$$

Algorithm 1: Training of Neural Network Using Deep Q-Network (DQN)

Initialize replay buffer B in Memory M ;
Initialize Q as a training network with θ_o random parameters ;
Initialize target network Q_n with random parameters $\theta_n = \theta_o$;
while *Episode = 1; Episodes < Total Episodes; Episodes++* **do**
  Start the Simulation with 1st Step J;
  foreach *J=1 to N* **do**
    Action = $\begin{cases} Arbitrary, & \text{with probability } \epsilon \\ arg\max'_a\ Q(s_o, a_o; \theta_o), & \text{otherwise} \end{cases}$
    Perform action a_n, and observe reward r, next state s_n;
    Save experiences (s_o, a_o, r, s_n) in B;
    Take mini batches from saved experiences as samples from replay buffer B ;
    Calculate the actual/targets L;
    L = $\begin{cases} 0, & \text{If terminated} \\ r + \max'_a\ Q_n(s_n, a_n; \theta_n), & \text{otherwise} \end{cases}$
    Do optimization as;
    $(L\text{-}Q(s_o, a_o;\theta_o))^2$ w.r.t θ;
    For every j' step ;
    Reset $Q_n = Q_o$;
    Set $s_o = s_n$;

where s and a are current state and action. During the training process, our agent gets the intersection's current state, the vehicles' action, and the reward of the recent action. All this data is stored in the replay buffer and used during the training. The agent explores the whole environment and builds complete steps for transitions for every action. At the start of the process, agent takes some actions randomly and some by looking in the Q- network, this behaviour of agent is known as an epsilon-greedy policy. We used loss function as the squared difference between actual and predicted value. We minimized the loss by updating the weights.

$$L = (r\ +\ \gamma max_{a'}\ Q\,(s_n,\ a_n\,;\ \theta_n)\ -\ Q(s_o, a_o;\ \theta_o))^2 \tag{5}$$

During the learning process, as we have no labels here so the actual value calculated from the $actual\ =\ R\ +\ \gamma\ \max_{a'}\ Q(s_o, a_o)$ and the $\max_{a'}\ Q(s_o, a_o)$ is the neural network's list of Q values of output data. Target network is also maintained here and actual weights are assigned to target network for every N number of steps. The Q-learning algorithm for our training process is given in algorithm 1

We did training by taking one sample from the batch of samples, and the data from sensors is fed to the input layer

of the neural network. We have 80 sensors for this process, so our neural network model has 80 neurons in the input layer. We fed the data from sensors to the input layer as a vector containing 80 values. Our output layer has only 4 neurons which correspond to the 4 Q-values associated with 4 possible actions as shown in figure 2.

The neural network is mostly known as universal function approximates as they can approximate any function. The network model's activation function gives it the non-linear properties and decides how a certain unit is activated. In this paper, we used a rectified linear activation function or ReLU, a linear function that directly outputs the result or makes it zero.

$$f(r) = \begin{cases} r, & \text{if } r \geq 1 \\ 0, & \text{otherwise} \end{cases}$$

We used Mean Squared Error and optimization for loss function by ADAptive Moment estimation (Adam) [44]. Adam is an optimization algorithm for stochastic objective function based on the first-order gradient.

5) EPSILON GREEDY POLICY

Initially, all the Q-values are zeroes. Then we iteratively update the Q values. As initially, all values are zero, then updating values for the first time we use epsilon. Epsilon is basically a trade-off between exploration and exploitation [45], [46]. Exploration is exploring the environment in which the agent is working. It will explore the environment regardless of the fear of higher negative rewards. The more the agent explores the environment, the more it learns about new states [47]. On the other hand, exploration is meant to reach the states based on the agent's knowledge based on epsilon values. Epsilon may also be called the probability to explore the environment. Initially, it means the agent will investigate the environment, and later it will decay at some rate, which means we are moving towards exploitation. Epsilon greedy policy generates a number between 0 and 1. If the random number is less than the epsilon value, we will select spontaneous action to explore the environment. On the other hand, if it is more significant than epsilon, we pick the best possible action based on the agent's knowledge. As the epsilon is decaying, the agent will move towards more exploitation rather than exploring new states. In other words, the agent is greedy about exploiting the current knowledge. The concept of max steps or episodes are the times the agent will play in the environment. The episode will end when the agent achieves its goal or reaches the max step's value. We are decaying the epsilon value episode by episode in the following way: $Epsilon = 1 - CurrentEpisodes/TotalEpisodes$

6) AGENT's MEMORY

An agent needs memory for learning and performance. For this purpose, we created a stack of samples. One sample consists of the old state, old action, reward, and current state as we need all these elements in our Bellman optimality equation. We have created a stack of 50,000 samples. If the stack is full, then we pop the oldest sample from the stack and insert the new sample.

The sample is composed of Previous states, Previous actions, Rewards, and Current state.

7) EXPERIENCE REPLAY

Experience replay strategy is used to update weights during the updating process. It chooses samples from memory explained in the previous section, which leads to efficient learning and optimized final policy. We took the batch size of 200 samples, selected randomly during our training process of neural networks. We are testing our agent on 100 episodes, and each episode has a maximum of 2500 steps (simulation steps).

We used this strategy because two consecutive states correlate with each other. An upcoming state that directly evolves from the previous state has chances of correlation and minimize this; we used experience replay (choosing randomized batches of samples from sample stack). Further, it will be helpful for our neural network to refresh the states which passed earlier.

Algorithm 2: Routing Vehicles to Alternate Paths

```
Initialize Memory ;
Calculate the O-D matrix for each vehicle ID;
Store Vehicle ID with O-D matrix in memory ;
Calculate shortest paths P1,P2 and P3 using Dijkstra
  algorithm;
Compute T-T (Travel-Time) for each route ;
Store the T-T against each path as P1-T-T, P2-T-T, and
  P3-T-T in route file;
foreach Vehicle at Intersection I do
    Get the current state of the intersection;
    Calculate T-W-T(Total-Wait-Time) on the
      intersection ;
    Add the T-W-T to the P1-T-T ;
    Calculate U-T-W-T(Updated-Total-Wait-Time) ;
    if (U-T-W-T > P2-TT or P3-TT) then Reroute
      Vehicles on the route P2 or P3;
    else if (U-T-W-T <= P2-T-T or P3-T-T) then
        Stay on the same route P1
```

When the DRL module is well trained by experience, it takes the current state of the intersection and calculates the queue length and waiting time of each vehicle. It checks for the appropriate action from replay memory. If the current action is not equal to the old action stored in the memory, it turns the yellow light. If the current action is equal to the old action, it turns on the green light. It means now our module is using a greedy policy (only best action) to optimize the traffic light. In the next section, we will see the working of our second module in detail.

B. I2V BASED SMART REROUTING

The traffic signal control described in the previous section solves the congestion problem at intersections by optimizing

traffic conditions. It minimizes the long queues and reduces the waiting time for vehicles at intersections. This approach could be further empowered by taking the higher-level view of the traffic situation and involving infrastructural components (I2V) because we want to improve traffic flow. The deep reinforcement module manages traffic on intersections, but what about the vehicles that have not yet reached the congested intersection. These vehicles can halt in the congestion for more longer than the vehicles reached the intersection, which could worsen traffic conditions. In this paper, we propose a smart routing technique to reroute the vehicles that are not yet reached on the intersection but coming towards it. The upcoming traffic is rerouted to alternate routes that are less congested but could be longer in travel times. For the purpose, we use lane area detectors in our experiments. We placed 16 detectors such that 4 detectors at each lane at each side of the intersection. These detectors give us the current view of the roads and the information about the routes. These detectors maintain the information about the traffic on each side in a log file separately. We analyze the data from detectors after every 30 seconds. A threshold limit is set for the intersection. If the number of vehicles exceeds the threshold limit and vehicles are coming continuously, then using that data from detectors about the number of vehicles, speed, and direction of vehicles, we shift traffic to alternate routes. This strategy holds equally good for signal-free routes and unique routes for VIP vehicles or ambulances. The combination of techniques provides a powerful metaphor to optimize the routes and avoid congestion. It also helps to achieve exception handling, e.g. when the traffic light fails, and cars are stuck near the intersection, it will help us by providing the information to reroute the upcoming traffic. The strategy ensures to reroute vehicles on the fastest route rather than the shortest congested route. For this purpose, first of all, we took the O-D (Origin-Destination) matrix against each vehicle's ID and computed all the paths from the start of a trip to the end. We calculated shortest paths using Dijkstra algorithm [48] and named them P1 for Path1, P2 for path 2, and so on. P1 is considered the shortest path, P2 as the second shortest path, P3 as the third shortest path, and so on. In normal conditions, AVs will take P1, which is the shortest path. Still, there is a possibility of a congested intersection on P1, and more vehicles are not allowed to enter the crowded intersection as it reached the threshold limit. Now, vehicles coming behind the intersection will have to decide whether they should wait behind the intersection or take alternate routes. In the case of autonomous vehicles, the decision will be based on the computation of travel times. The Total-Wait-Time (T-W-T) on the intersection is added to the travel time of Path 1 (P1-T-T), and an Updated-Total-Wait-Time (U-T-W-T) is obtained. If U-T-W-T of P1 is greater than (P2-T-T), than the vehicle is rerouted on P2. If U-T-W-T is less than or equal to P2-T-T, than the vehicles waits for the intersection. In the simulation setup, we are assuming that P2 is the signal-free route. Computations become less complicated; therefore, we are taking one shortest path, P1 and two alternate paths P2 and P3. Algorithm2 provide a snapshot of the steps involved in the process of rerouting the vehicles.

V. EXPERIMENTAL SETUP

In this section, the experimental setup is discussed in detail. We conducted our experiments in the simulation tool SUMO (Simulation of Urban MObility) [49]. SUMO is an open-source traffic simulator that provides simulation of traffic in real-time. The Python-based API provided by SUMO manages the timings of traffic signals. Description of vehicles, roads, traffic sensors is given in detail in the following subsections.

A. PROPERTIES OF INFRASTRUCTURAL NETWORK

A detailed description of road network properties, traffic policy, and sensors is given with other necessary information in this subsection.

1) ROADS AND LANES USED

The road network used for simulation is shown in Fig. 3. It becomes complex to manage the sensor's data, detector's data, and traffic flow as we try to get optimum performance from agents since the computational cost is high. There are multiple paths to reach from one point to another point in the network. Some tracks have traffic signals, and some are signal-free, but as mentioned earlier, signal-free paths are usually lengthier. We have chosen the roads' length: the South to the intersection, north to the intersection, west to the intersection, and east to the intersection (with traffic signal) as 800 (750+50) meters. The length of the roads on the diagonals is 1040 meters each. Signal free paths are available through roads on diagonals. All the roads connected to traffic signal are 4 lane roads (4 lanes for incoming and 4 lanes for the outgoing). Two middle lanes on each side are for those vehicles that want to travel straight, and one lane is for those vehicles which wish to turn left or right on each side of the intersection.

Simulation Parameters	
Parameter	Value
Simulator	SUMO
Number of Vehicles per Episode	1000
Number of traffic lights	01
Simulation Time	2500 (s)
Simulation Time for each vehicle	41 (min)
Simulation Map	4 leg intersection
Transmission Range	750 (m)

2) ROUTE ASSIGNMENT IN NETWORK

There are two types of route assignment. First is static that is set before simulation or initially assigned routes. Second is the dynamic route assignment implemented with the help of TraCI based on the run time traffic situation on roads. In static route assignment, 60 per cent of vehicles are allowed to go on

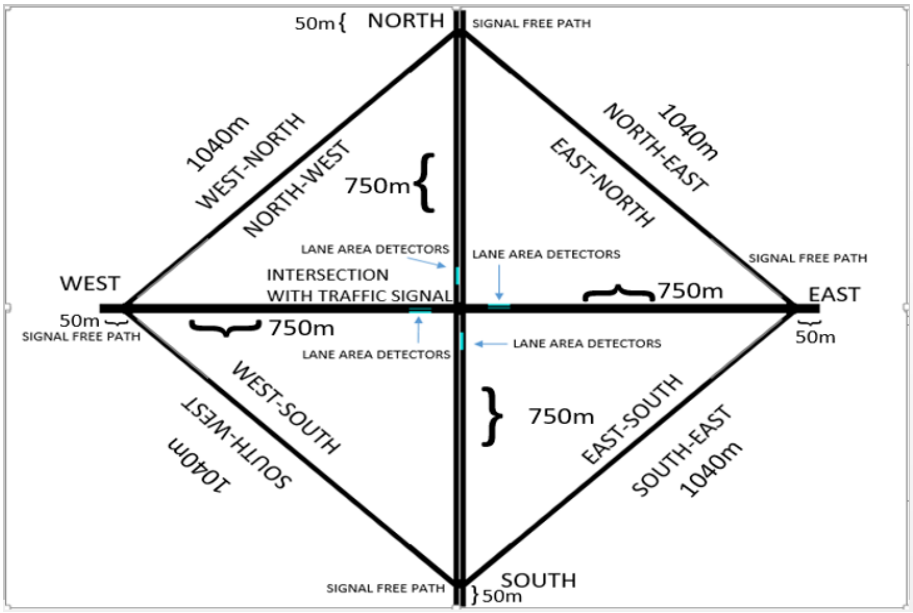


FIGURE 3. Road network configuration.

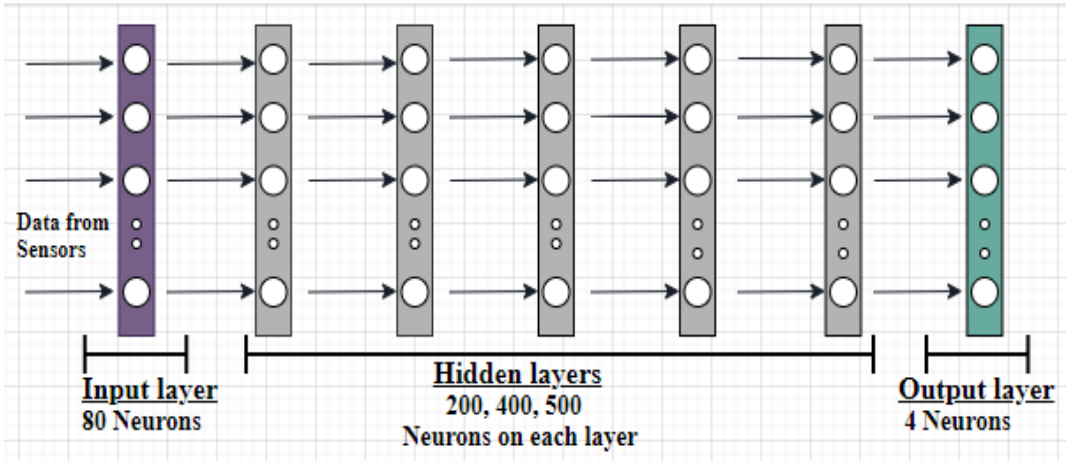


FIGURE 4. Neural network models for deep reinforcement learning based traffic light control. Models has same input(80 Neurons), output(4 Neurons), but different hidden layer's neurons (400 for Model 1, 500 for Model 2 and 200 for Model 3).

the straight path (which will pass through the intersection), and 40 per cent will follow directions using a left turn or right turn lane, signal-free trails, and alternate paths. Traffic condition is constantly monitored by the detectors placed on east, north, west, and south regions. The log file of data obtained from detectors is preserved and maintained of each side separately. If vehicles are more than the threshold limit specified on the road based on its capacity, the message is passed to upcoming vehicles whether to wait or to take alternate routes based on the rerouting strategy.

3) TRAFFIC GENERATION

We tested different models of the neural network on almost 1000 vehicles. Every second, when a new vehicle joins the network, a route is assigned to it based on its destination. We used different vehicles (standard cars, buses, trailers, and ambulances) for our simulation having different speeds. Each vehicle takes route according to its destination, which is most probably the shortest route. In the simulation setup, 60 per cent of the traffic will take the direct shortest path from origin to destination lying opposite to each other while Each vehicle having to pass through the signal intersection. Remaining 40 per cent traffic may or may not have to pass through the signalled intersection. These vehicles may take the alternate path that is a signal-free path, which in some cases becomes the fastest path when it comes to time calculation. For example, if a vehicle wants to move from west to east, it has a direct shortest signalled path of length 1040 meters, and if it follows the path via signal-free intersection, it needs to travel 1500+ meters which is not the shortest path but fastest path that takes less time due to the congestion on signalled path.

4) WEIBULL PROBABILITY DENSITY FUNCTION

Vehicles at the intersection can be defined by modeling vehicles' arrival time at that intersection in the given interval of time. It could be done by specifying the time interval between two successive vehicles arriving at the intersection. We used the well-known and one of the most widely used existing probability distribution, i.e., Weibull distribution to model the vehicles at the intersection. Probability Density Function equation is usually used to express Weibull distribution. The three-parameter Weibull distribution is the most general expression available in the literature, and is given below:

$$f(T) = \frac{\beta}{\eta}\left(\frac{T-\gamma}{\eta}\right)^{\beta-1} e^{-\left(\frac{T-\gamma}{\eta}\right)^{\beta}} \tag{6}$$

Training Parameters and attributes of agent	
Parameter	Value
GUI	False
Total Episodes	100
Gamma Value	0.5
Batch Size	200
Memory Size	5000
Number of states	80
Actions	4
Maximum Step	2500
Green Duration	5(s)
Yellow Duration	2(s)

β, η and γ are the parameters for slope, scale and location respectively. We need only scale parameter for vehicles' arriving time, so by setting $\gamma = 0$ and assuming $\beta = C = Constant$ our distribution become one parameter weibull.

Fig.5 is the visualization of weibull distribution which shows the heavy and light traffic volumes during congestion and normal situations.

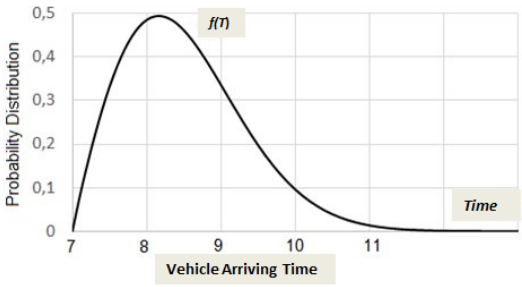


FIGURE 5. Weibull distribution for vehicle arrival time.

5) NETWORK OF SENSORS

We use sensors for obtaining current traffic information. The simulation used 80 sensors that are spread on different places on the incoming traffic lanes, 20 sensors on each intersection side. On each side of intersection, 10 sensors are placed on 3 lanes, i.e. 2 straight lanes and right turn lane because all of these three lanes share the same traffic light. The remaining 10 sensors are placed on the left turning lane. These sensors are placed at different distances from the traffic light. These sensors will provide information about vehicles' position, which could be, e.g., 10 wheeler trailers, school bus, emergency car, standard cars, or VIP cars on the lanes. All 80 sensors have a unique identity, and the density of sensors close to the traffic light is high (as it is of more importance) than at the end of the lane. If at least one vehicle exists in the sensor's vicinity, it will give a signal about its presence. In this way positions of all the vehicles could be tracked which are approaching traffic signals. Now there are 2^{80} possible different states in the network. This 2^{80} conditions are computationally challenging to manage for the agent.

Structure of Neural Network	
Parameter	Value
Simulator	SUMO
Neural network layers	200, 400, 500
Loss Function	Mean Squared Error
Activation Function	Relu
Optimization Function	Adam Optimizer

6) PERFORMANCE PARAMETERS

These are the underneath performance parameters used for performance evaluation

a: REWARD

The agent should take actions that maximize the reward. The general trend of all the networks is that the negative reward approaches zero, which means the agent learns gradually.

b: DELAY

When cars are stuck in traffic jams, they have to wait more and as a result, delay increases. The cumulative delay (summation of all cars' delay) decreases gradually when the agent gets to know about the environment.

c: QUEUE LENGTH

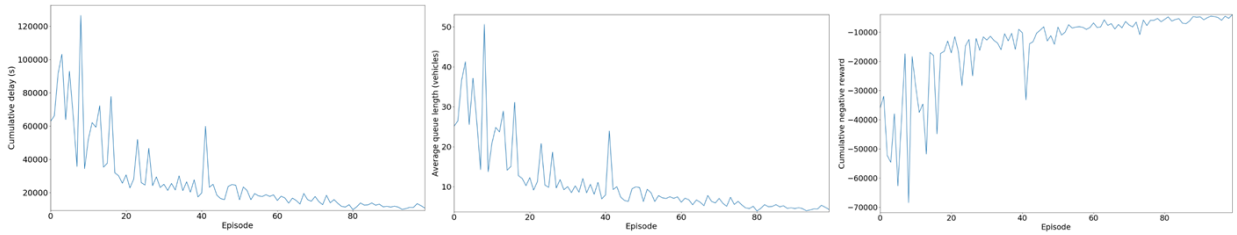
When roads are loaded with traffic and signals are closed during peak hours, queues of vehicles waiting for a signal to turn on (green) become lengthy. We selected the average queue length as the performance parameter of the agent. It should decrease with the increase in episodes.

d: SIMULATION TIME

When the last car exits the simulation environment, the simulation ends. The maximum time for simulation is 2500 seconds which is more than half an hour. Suppose we reduce congestion with the help of reinforcement learning and I2V communication. In that case, vehicles have to wait for less, and they will reach their destinations quicker, and eventually, the simulation time will reduce.

VI. RESULTS AND DISCUSSIONS

We have conducted three different steps in experiments. In the first step, we ran traffic typically using traditional settings (using fixed time signals) without implementing reinforcement learning or smart routing and observed the traffic



A:Cumulative Delay Model 1 *B:Average Queue Length Model 1* *C:Cumulative Negative Reward Model 1.*

FIGURE 6. Cumulative delay, average queue length for model 1 with 400 neurons, start decreasing while total negative reward starts increasing as episodes are increasing.

condition. We recorded these experiments' results for the comparisons after applying our deep reinforcement learning based solution and smart routing solution. In the second step of the experiments, we implemented a deep reinforcement learning module to make intelligent traffic light signals to optimize traffic on intersections and recorded the results. In the third step, using an intelligent traffic light module on the intersection, we implemented a smart routing module to reroute the traffic headed towards the intersection, load-balanced the traffic flow and observed our approach's final results. We further implemented our strategy by changing our neural network model's number of neurons, changing the gamma values, and showing its effects. Then we performed experiments by changing the number of hidden layers of the neural network model and proved with the results why we used a 5-layered neural network model for our approach.

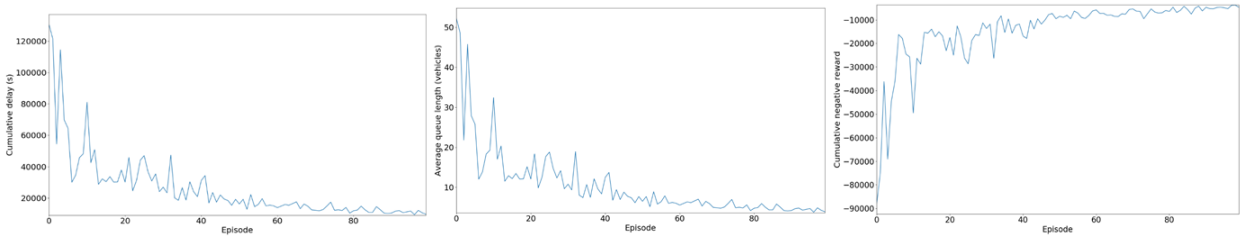
The neural network model we used for our experiments is shown in Fig.4. The neural network has 80 states as input and 4 states at the output layer, and we changed the number of neurons in the hidden layers to 200, 400 and 500. There is no hard and fast rule in deep learning on choosing the number of hidden layers or the number of neurons on hidden layers. Selecting the number of hidden layers and the number of neurons in the hidden layer is an integral part of planning and design of neural network architecture. Even though these layers do not interact directly with the external environment, but these layers have significant impact on the output. Similarly, using too many neurons in layers could cause over-fitting, which means that our model has enormous information processing capacity but a limited amount of information. Using too few neurons could cause under-fitting, which means that significantly fewer neurons are available to detect the correct signal in massive data set adequately. Therefore we should carefully consider these layers. We chose a 5-layer network because it worked best for our case, and we proved it by results. Now we will discuss in detail the quantitative effects of our two-phased approach.

A. WHEN NO REINFORCEMENT AND SMART ROUTING TECHNIQUE APPLIED

We used 1000 vehicles for our experiments. We have no intelligent traffic light controller implemented on the intersection, or no routing technique is applied at this step of experiments. We implemented simple pre-timed traffic light at the intersection and observed the behaviour of vehicles. In a few seconds, we observed that vehicles got halted in the congested intersection, and there were long queues of cars at the intersection and behind the intersection. We used the same road network, as explained in the previous section, for all experiments. Results of 1st step of our experiments in Fig.10 green bar show that 1000 vehicles are taking almost 2500 seconds to get out of the specified area on the road network due to the congestion. Now we will see the implementation of our approach and results in the next sub-sections.

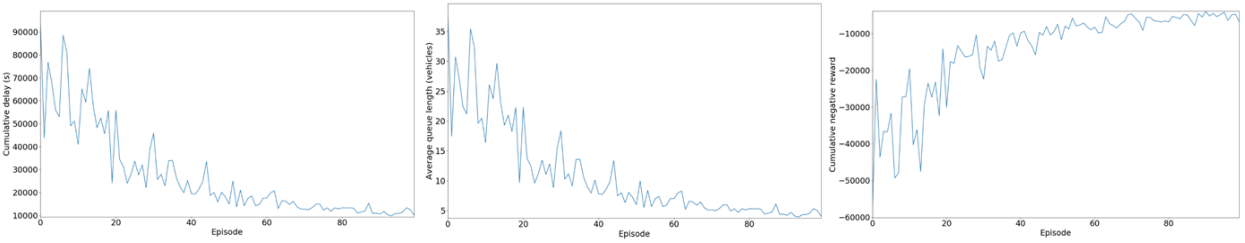
B. WHEN INTELLIGENT TRAFFIC LIGHT MODULE IMPLEMENTED

In the second step of the experiments, we implemented intelligent traffic light control using deep reinforcement learning to control the traffic and schedule traffic lights. Our intelligent traffic light module takes the current state of the intersection from sensors, compute the waiting times (delay) of vehicles, measures the queue lengths on the intersection, and takes the appropriate actions (turn on the appropriate lights) situation. Our traffic light agent aims to minimize cumulative delay, average queue length, and negative reward. We explore results and find a significant reduction in waiting time and the longest queues in congested environments. There is a general trend in all the models, i.e. with the passage of episodes, the cumulative delay and average queue length decrease while the agent's reward increases (total negative reward tends towards 0). We are taking a neural network model with 400 layers as model 1, with 500 layers as model 2, and 200 layers as model 3. The part A of Fig. 6, 7 and 8 shows the cumulative delay or waiting time of model 1, 2 and 3; vehicles tend to decrease as episodes increases and agents start to learn things. At each episode, we test the average delay time of vehicles. Traffic rates on all lanes are also the same. Similarly, the part B of Fig.6, 7 and 8 shows the average queue lengths of vehicles are also decreasing as the agent starts learning about the environment in a better way and congestion is reducing, making the overall traffic flow efficient. Total negative reward in part C of 6, 7 and 8 starts increasing, which means the agent is performing better with the increment of episodes. The blue bar in Fig.10 shows that the average waiting time of vehicles reduced from 2500(s) to 1980(s) when applied to the neural network model with 200 neurons, to 2180(s) when applied to the model with 400 layers and 2200(s) when applied on model with 500 layers. Here we can



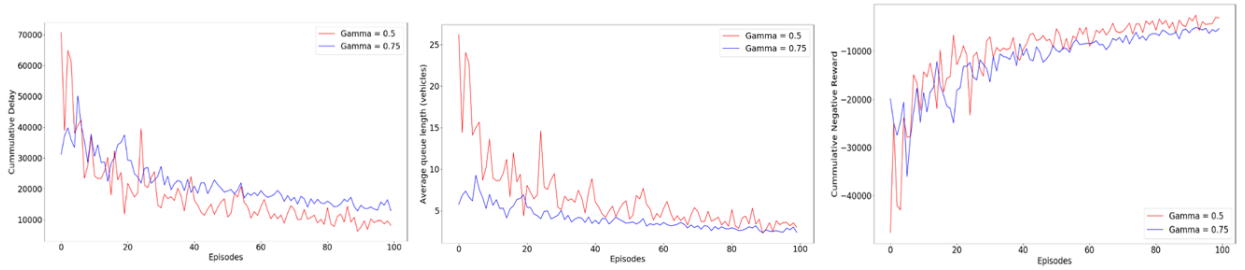
A:Cumulative Delay Model 2 *B:Average Queue Length Model 2* *C:Cumulative Negative Reward Model 2*

FIGURE 7. Cumulative delay, average queue length for model 2 with 500 neurons, start decreasing while total negative reward starts increasing as episodes are increasing.



A:Cumulative Delay Model 3 *B:Average Queue Length Model 3* *C:Cumulative Negative Reward Model 3*

FIGURE 8. Cumulative delay, average queue length for model 3 with 200 neurons, start decreasing while total negative reward starts increasing as episodes are increasing.



A:Cumulative Delay *B:Average Queue Length* *C:Cumulative Negative Reward*

FIGURE 9. Changing gamma value for cumulative delay, average queue length and negative reward.

notice that the performance degradation of neural network models increases the number of neurons on hidden layers. Specifying the best values for any neural network model can be done via systematic experimentation, and the neural network model with 200 layers is performing best for our system. An apparent reduction of time with DRL controlled traffic lights reduced the congestion level by 13%.

C. WHEN INTELLIGENT TRAFFIC LIGHT MODULE WITH SMART REROUTING MODULE IMPLEMENTED

In the third set of experiments, we applied Smart Rerouting (SR). The RL module controls congestion at the intersection and controls the traffic coming behind the intersection. We implemented the smart rerouting technique to reroute vehicles to alternate paths so that the traffic load on the intersection could be distributed among other routes, and vehicles could avoid congested situations. The red bar in Fig 10 shows more reduction of time by 12%. The total decrease in time by using both intelligent traffic lights and smart routing module is 25%. We tested all three sets of experiments on neural network model two and three to check the model's performance.

D. CHANGING THE NUMBER OF NEURONS OF NEURAL NETWORKS

We have tested our approach on different parameters by changing the number of neurons in the hidden layers. The Fig.10 shows the difference in the results of neural networks with different neurons. The blue bar of the performance graph with 200 neurons shows that RL controlled traffic lights helped reduce simulation time by around 17%, and SR module further reduced simulation time by around 14%. The total reduction in simulation time is 31% in this case. In Fig.10, we will observe that when we increased the number of neurons to 400 the RL controlled traffic lights helped reduce simulation time by around 12% and SR components further reduced simulation time by around 9%. The total reduction in simulation time is 21%. Similarly, when we increased the neurons to 500 the total reduction is 12%. As we expand the number of neurons on neural networks, our performance degrades, so we can say that our approach's best model is when the number of neurons is 200 as it reduces the simulation time up to 31%.

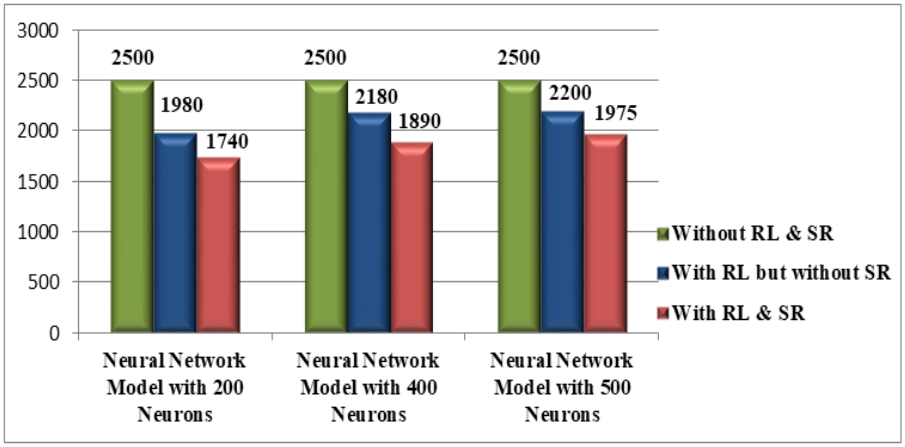


FIGURE 10. Performance graph when no Reinforcement Learning(RL) or Smart Routing(SR) applied, when only RL applied, When RL with SR applied.

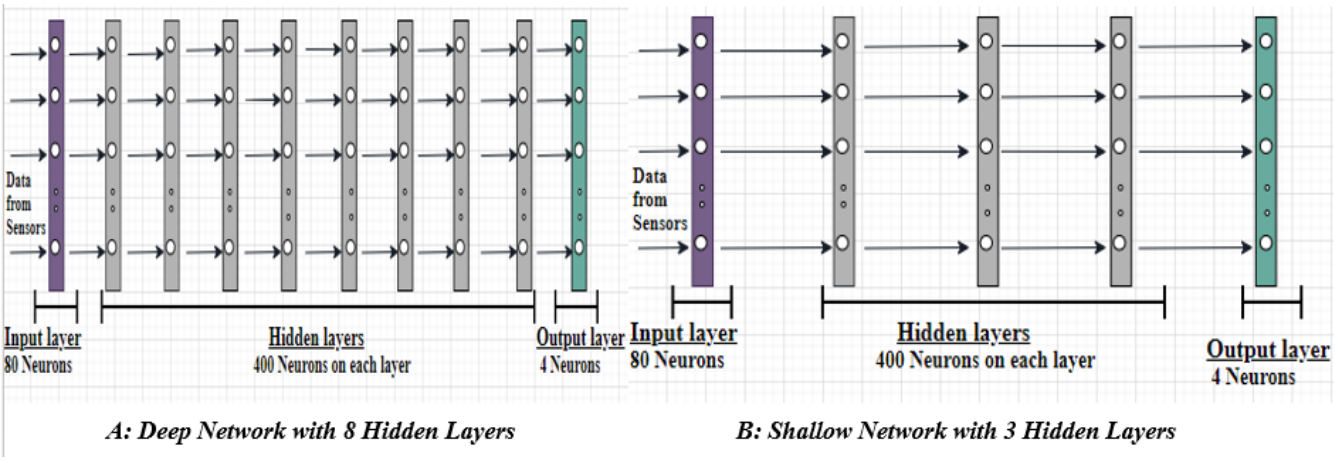


FIGURE 11. Deep network with eight hidden layers vs shallow network with three hidden layers.

E. CHANGING GAMMA VALUE

We changed the gamma value (discount factor) to check the effects on our neural network model. The blue line shows gamma value $=$ 0.5, whereas the red line shows gamma value $=$ 0.75. In Fig.9-A,B and C, we can notice a slight difference in output by changing gamma value. The performance with gamma value equals to 0.5 is slightly better than with gamma value equals to 0.75.

F. DEEPER VS SHALLOWER NETWORKS

In the final part, we checked what happens if we make our neural network more deep or shallow (increasing or decreasing hidden layers). We experimented by varying the number of hidden layers for our neural network. A neural network with eight hidden layers is shown in Fig.11-A, whereas a shallower network with three hidden layers is depicted in Fig.11-B. The reason to do experiments using more and few hidden layers is that using few hidden layers is sufficient for some functions, but for others, the number of hidden layers is less than the system becomes inefficient. The performance graph in Fig.12 shows that simulation time increased by 21% as compared to our best model. Similarly, the performance graph with an external network shows an increment by 18%. We can see the effect of performance degradation by adding or removing layers from Neural Network.

Since traffic data is non-linear and dynamic, we use systematic experimentation to determine what works best for our specific model. We perform experiments with varying number of hidden layers and also neurons, and analyze accuracy of epochs. Less number of layers causes underfitting, and more number of layers results in over-fitting. Finally, we choose 5 layers neural network, which gives the best results for our specific problem. All the results show that using reinforcement learning to control traffic light and infrastructural components for smart routing can improve performance and make traffic flow efficient in autonomous vehicles. The change in neurons in hidden layers can also affect the neural network's performance, and changing the number of hidden layers can also affect the traffic flow's performance. We have tuned an optimum model that performs better through these experiments than others to increase traffic efficiency. When combined with this model with the routing module, we obtain the best performance in the given

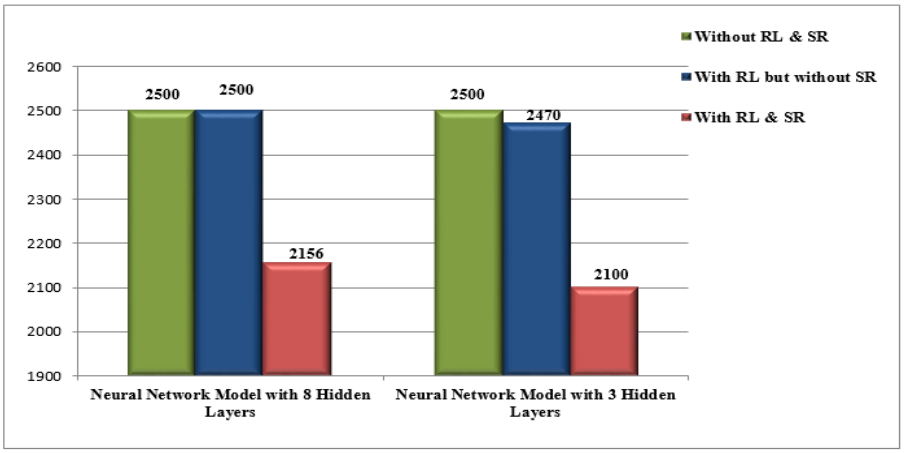


FIGURE 12. Performance of deep vs shallow neural networks.

scenarios. All experiments suggest that using a combination of strategies, specifically in traffic scenarios, gives better results than just a single technique. Deep reinforcement learning gave us optimal traffic control on the intersection while rerouting managed traffic behind the intersection, maintaining the overall traffic flow that directly leads to safety management, fuel consumption reduction, and accidents avoidance.

VII. CONCLUSION

We present a two-phased approach to control the traffic and avoid congestion in scenarios where all vehicles behave autonomously. In the first phase, we build a deep reinforcement learning-based intelligent traffic light module to optimize traffic flow during congestion. We observe that traffic signals are using longer queue lengths and waiting times to adjust dynamically. We demonstrated that using a 5 layered deep neural network model with 200 neurons on the hidden layers optimizes our model's performance and gives the best solutions through extensive simulation results. In the second phase of our approach, we proposed a smart rerouting technique that considers the congestion on the intersection by adjusting a thresh-hold limit and calculating travel times for each vehicle's O-D matrix. We compared the distance of the current route and waiting time on the intersection with the alternate routes; it reroutes the vehicles to the roads, which takes less time. Results show significant improvement in congestion situation and in managing traffic flow. Experiments show the effectiveness of our approach compared to the traditional settings; 17% reduction of time is achieved when intelligent traffic light module is applied, and 14% more reduction in time is obtained when the smart rerouting module is implemented. The results show the total performance efficiency up to 31%.

In future, we plan to extend this approach to more complex road networks, with multiple signal intersections, and real-world road data set. Including V2V coordination with V2I and I2V could lead us to better traffic flow management for future smart cities.

REFERENCES

[1] L. Figueiredo, I. Jesus, J. A. T. Machado, J. R. Ferreira, and J. L. M. D. Carvalho, "Towards the development of intelligent transportation systems," in *Proc. ITSC. IEEE Intell. Transp. Syst.*, Aug. 2001, pp. 1206–1211.

[2] G. E. Cantarella, "Day-to-day dynamic models for intelligent transportation systems design and appraisal," *Transp. Res. C, Emerg. Technol.*, vol. 29, pp. 117–130, Apr. 2013.

[3] J. K. Hedrick, M. Tomizuka, and P. Varaiya, "Control issues in automated highway systems," *IEEE Control Syst.*, vol. 14, no. 6, pp. 21–32, Dec. 1994.

[4] S. El Hamdani and N. Benamar, "Autonomous traffic management: Open issues and new directions," in *Proc. Int. Conf. Sel. Topics Mobile Wireless Netw. (MoWNeT)*, Jun. 2018, pp. 1–5.

[5] T. Litman, *Autonomous Vehicle Implementation Predictions*. Victoria, BC, Canada: Victoria Transport Policy Institute, 2017.

[6] R. Arnott and K. Small, "The economics of traffic congestion," *Amer. Sci.*, vol. 82, no. 5, pp. 446–455, 1994.

[7] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transp. Res. A, Policy Pract.*, vol. 94, pp. 182–193, Dec. 2016.

[8] M. V. Rajasekhar and A. K. Jaswal, "Autonomous vehicles: The future of automobiles," in *Proc. IEEE Int. Transp. Electrific. Conf. (ITEC)*, Aug. 2015, pp. 1–6.

[9] H. Hashim and M. Omar, "Towards autonomous vehicle implementation: Issues and opportunities," *J. Soc. Automot. Eng. Malaysia*, vol. 1, no. 2, pp. 111–123, 2017.

[10] V. Milanes, J. Villagra, J. Godoy, J. Simo, J. Perez, and E. Onieva, "An intelligent V2I-based traffic management system," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 49–58, Mar. 2012.

[11] S. Campbell, N. O'Mahony, L. Krpalcova, D. Riordan, J. Walsh, A. Murphy, and C. Ryan, "Sensor technology in autonomous vehicles: A review," in *Proc. 29th Irish Signals Syst. Conf. (ISSC)*, Jun. 2018, pp. 1–4.

[12] X. Li, X. Xu, and L. Zuo, "Reinforcement learning based overtaking decision-making for highway autonomous driving," in *Proc. 6th Int. Conf. Intell. Control Inf. Process. (ICICIP)*, Nov. 2015, pp. 336–342.

[13] J. R. N. Forbes, *Reinforcement Learning for Autonomous Vehicles*. Berkeley, CA, USA: Univ. of California, 2002.

[14] X. Liang, X. Du, G. Wang, and Z. Han, "Deep reinforcement learning for traffic light control in vehicular networks," 2018, *arXiv:1803.11115*. [Online]. Available: http://arxiv.org/abs/1803.11115

[15] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 2641–2646.

[16] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," 2016, *arXiv:1610.03295*. [Online]. Available: http://arxiv.org/abs/1610.03295

[17] A. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electron. Imag.*, vol. 2017, no. 19, pp. 70–76, Jan. 2017.

[18] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, Dec. 2020.
[19] S. H. Khan, A. Sohail, and A. Khan, "COVID-19 detection in chest X-ray images using a new channel boosted CNN," 2020, *arXiv:2012.05073*. [Online]. Available: http://arxiv.org/abs/2012.05073
[20] F. L. Hall, "Traffic stream characteristics," in *Traffic Flow Theory: State-of-the-Art Report*, N. Gartner, C. J. Messer, and A. K. Rathi, Eds. Washington, DC, USA: FHWA/TRB/ORNL, 2001, ch. 2.
[21] S. Maerivoet and B. D. Moor, "Traffic flow theory," 2005, *arXiv:physics/0507126*. [Online]. Available: https://arxiv.org/abs/physics/0507126
[22] T. Brys, T. T. Pham, and M. E. Taylor, "Distributed learning and multi-objectivity in traffic light control," *Connection Sci.*, vol. 26, no. 1, pp. 65–83, Jan. 2014.
[23] Q. Lin, B. W. Kwan, and L. J. Tung, "Traffic signal control using fuzzy logic," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. Comput. Cybern. Simulation*, vol. 2, Oct. 1997, pp. 1644–1649.
[24] M. Chee Choy, D. Srinivasan, and R. Long Cheu, "Cooperative, hybrid agent architecture for real-time traffic signal control," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 33, no. 5, pp. 597–607, Sep. 2003.
[25] C. Cai, C. K. Wong, and B. G. Heydecker, "Adaptive traffic signal control using approximate dynamic programming," *Transp. Res. C, Emerg. Technol.*, vol. 17, no. 5, pp. 456–474, Oct. 2009.
[26] P. Mannion, J. Duggan, and E. Howley, "An experimental review of reinforcement learning algorithms for adaptive traffic signal control," in *Autonomic Road Transport Support Systems*. Berlin, Germany: Springer-Verlag, 2016, pp. 47–66.
[27] K. Park and W. Willinger, *Sele-Similar Network Traffic and Performance Evaluation*. Hoboken, NJ, USA: Wiley, 2000.
[28] C. Gershenson, "Self-organizing traffic lights," 2004, *arXiv:nlin/0411066*. [Online]. Available: https://arxiv.org/abs/nlin/0411066
[29] M. Abdoos, N. Mozayani, and A. L. C. Bazzan, "Hierarchical control of traffic signals using Q-learning with tile coding," *Int. J. Speech Technol.*, vol. 40, no. 2, pp. 201–213, Mar. 2014.
[30] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *J. Transp. Eng.*, vol. 129, no. 3, pp. 278–285, May 2003.
[31] S. El-Tantawy and B. Abdulhai, "An agent-based learning towards decentralized and coordinated traffic signal control," in *Proc. 13th Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2010, pp. 665–670.
[32] M. Wiering, "Multi-agent reinforcement learning for traffic light control," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 1151–1158.
[33] M. Steingrover, R. Schouten, S. Peelen, E. Nijhuis, and B. Bakker, "Reinforcement learning of traffic light controllers adapting to traffic congestion," in *Proc. BNAIC*, 2005, pp. 216–223.
[34] J. Iša, J. Kooij, R. Koppejan, and L. Kuijer, "Reinforcement learning of traffic light controllers adapting to accidents," in *Proc. Design Organisation Autonomous Syst.*, 2006, pp. 1–14.
[35] M. Abdoos, N. Mozayani, and A. L. C. Bazzan, "Traffic light control in non-stationary environments based on multi agent Q-learning," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2011, pp. 1580–1585.
[36] S. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," *IET Intell. Transp. Syst.*, vol. 11, no. 7, pp. 417–423, Sep. 2017.
[37] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intell. Transp. Syst.*, vol. 4, no. 2, pp. 128–135, 2010.
[38] H. M. A. Aziz, F. Zhu, and S. V. Ukkusuri, "Learning-based traffic signal control algorithms with neighborhood information sharing: An application for sustainable mobility," *J. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 40–52, Jan. 2018.
[39] C. Wuthishuwong and A. Traechtler, "Distributed control system architecture for balancing and stabilizing traffic in the network of multiple autonomous intersections using feedback consensus and route assignment method," *Complex Intell. Syst.*, vol. 6, pp. 165–187, Nov. 2019.
[40] H. Schepperle and K. Böhm, "Agent-based traffic control using auctions," in *Proc. Int. Workshop Cooperat. Inf. Agents*. Berlin, Germany: Springer-Verlag, 2007, pp. 119–133.
[41] X. Dai, C. K. Li, and A. B. Rad, "An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 3, pp. 285–293, Sep. 2005.
[42] L. Shoufeng, L. Ximin, and D. Shiqiang, "Q-learning for adaptive traffic signal control based on delay minimization strategy," in *Proc. IEEE Int. Conf. Netw., Sens. Control*, Apr. 2008, pp. 687–691.
[43] X. Xu, L. Zuo, and Z. Huang, "Reinforcement learning algorithms with function approximation: Recent advances and applications," *Inf. Sci.*, vol. 261, pp. 1–31, Mar. 2014.
[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[45] V. Derhami, V. J. Majd, and M. N. Ahmadabadi, "Exploration and exploitation balance management in fuzzy reinforcement learning," *Fuzzy Sets Syst.*, vol. 161, no. 4, pp. 578–595, Feb. 2010.
[46] M. Coggan, "Exploration and exploitation in reinforcement learning," School Comput. Sci., McGill Univ., Montreal, QC, Canada, Tech. Rep. 1, 2004.
[47] S. B. Thrun, "Efficient exploration in reinforcement learning," Tech. Rep., 1992.
[48] M. Noto and H. Sato, "A method for the shortest path search by extended Dijkstra algorithm," in *Proc. SMC Conf. IEEE Int. Conf. Syst., Man Cybern. Cybern. Evolving Syst., Hum., Organizations, Complex Interact.*, vol. 3, Oct. 2000, pp. 2316–2320.
[49] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of sumo-simulation of urban mobility," *Int. J. Adv. Syst. Meas.*, vol. 5, nos. 3–4, 2012.

ANUM MUSHTAQ is currently pursuing the Ph.D. degree in traffic management, autonomous vehicles with the Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, Pakistan. Her research interests include traffic flow management, autonomous vehicles, reinforcement learning, the Internet of Things, industry 4.0, and cloud computing.

IRFAN UL HAQ received the M.Sc. degree in physics from Government College University, Lahore, Pakistan, and the Ph.D. degree in cloud computing from the University of Vienna, Austria. He is currently a Principal Scientist with the Pakistan Institute of Engineering and Applied Sciences. His research interests include blockchain, cloud computing, the Internet of Things, and autonomous vehicles.

MUHAMMAD USMAN IMTIAZ is currently an Alumnus of the Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, Pakistan. He is also working as an Associate Data Scientist with i2c Inc. His research interests include NLP, deep learning, reinforcement learning, and text mining.

ASIFULLAH KHAN received the M.S. and Ph.D. degrees in computer systems engineering from the GIK Institute. He is currently a Professor with the Pakistan Institute of Engineering and Applied Sciences. His research interests include machine learning, pattern recognition, image processing, computational intelligence, and deep neural networks.

OMAIR SHAFIQ is currently an Assistant Professor with the School of Information Technology, Carleton University, Canada. His research interests include data modeling, big data analytics, services computing, machine learning, and cloud computing.