Deep Reinforcement Learning-Based Traffic Light Scheduling Framework for SDN-Enabled Smart Transportation System

Neetesh Kumar, *Member, IEEE*, Sarthak Mittal, Vaibhav Garg, and Neeraj Kumar, *Senior Member, IEEE*

***Abstract*— This work proposes a traffic-light scheduling framework using the deep reinforcement learning technique to balance the traffic flow and to prevent congestion in the dense regions of the city via a software-defined control interface. A software-defined control enabled architecture is proposed to monitor the traffic conditions and it generates the traffic light control signal (Red/Yellow/Green) accordingly. For an intelligent traffic light control signal, a Deep Reinforcement Learning (DRL) model is proposed which takes vehicular dynamics as inputs from the real-time traffic environment such as heterogeneous vehicles count, speed, traffic density etc. To determine the congestion, a threshold policy is proposed and deployed on control server which generates the congestion prevention signal. A DRL agent operates in the coordination of congestion prevention signal and generates an effective traffic light control signal. The proposed model is evaluated through a realistic simulation on Indian city OpenStreetMap by using a well-known open-source simulator (SUMO). The comparative results show that the proposed solution improves several performance metrics such as average waiting time, throughput, average queue length, and average speed in the interval of 28.34% – 66.62%, 24.76% – 66.60%, 30.89% – 69.80%, and 16.62% – 43.67% respectively over other states of the art approaches.**

***Index Terms*— Intelligent transportation system, traffic light control system, deep reinforcement learning (DRL), software defined networking (SDN), scheduling.**

I. Introduction

DYNAMIC Traffic Light Control Systems (DTLCSs) are being practiced around the globe due to enormous growth in sensing, computing, and communication transport infrastructure such as sensors, Road Side Units (RSUs), traffic surveillance cameras, On-Board Units (OBUs) [1]. With these components, DTLCSs capture the real-time traffic state and take the decision about the traffic light control phase, i.e., red/green/yellow at the respective intersection. DTLCS indeed plays a significant role in minimizing the queue length, waiting time, wastage of fuel, and congestion in the city. Though, continuous growth in vehicles and limited road infrastructure cause congestion in many regions of the city. For such scenarios, Fixed Traffic Light Control System (FTLCS) is not effective to prevent congestion and to maintain a smooth traffic flow. Such FTLCSs are paralyzed with an increased inflow rate of the vehicles and relatively lower outflow rate at certain point of times. Thus, highly congested routes transform into traffic jams which results in the delay of many normal/priority/emergency vehicular services. To prevent such chaotic traffic situations, DTLCS is required to act intelligently via regular monitoring and learning the vehicular dynamics in order to make the traffic light control decisions effectively.

The use of intelligent techniques in the transportation system has significantly elevated social expectations including individual travellers due to promised solutions claimed by the underlying technologies. Following this, Reinforcement Learning (RL) technique is used to design intelligent traffic light controllers [2]. To design such systems, sensors visualize the real-time traffic and a controller to process the traffic data are the key components. Recent advances in sensors, cameras, and networking technologies enabled the capturing real-time traffic states such as vehicle location, vehicle count, speed, etc. [3]. RL is used as a key machine learning technique for the traffic light controller design with an aim of training an agent to learn optimal policy while interacting with the environment in order to maximize some reward [4]. Though capturing accurate real-time traffic state is not always possible, and the application of the RL technique to generate efficient traffic light control signals becomes more complicated as the number of traffic states grows exponentially. Therefore, researchers explore in utilization of deep neural networks to deal with a large number of traffic states [5].

Traditionally used centralized or decentralized control architectures are not sufficient enough to offer cost-effective, scalable, and manageable solution. Since centralized architecture is not scalable, and it suffers from a single point of failure, and decentralized architecture offers scalable architecture but difficult to manage as it requires high maintenance and operational costs. Therefore, software-defined wireless sensors network, in traffic management, has been progressed as a

Manuscript received January 27, 2021; revised May 13, 2021; accepted June 30, 2021. Date of publication August 11, 2021; date of current version March 9, 2022. This work was supported by the Science and Engineering Research Board (SERB), Department of Science and Technology (DST), Government of India, under the Project through Software Defined Controlled and Dynamic Traffic Load Balanced Scheduling Framework for the IoT Enabled Intelligent Transportation System (ITS) under Grant EEQ/2019/000182. The Associate Editor for this article was T.-H. Kim. *(Corresponding author: Neetesh Kumar.)*

Neetesh Kumar is with the Indian Institute of Technology Roorkee, Roorkee 247667, India (e-mail: neetesh@cs.iitr.ac.in).

Sarthak Mittal and Vaibhav Garg are with the ABV-Indian Institute of Information Technology and Management Gwalior, Gwalior 474015, India.

Neeraj Kumar is with the Thapar Institute of Engineering and Technology, Patiala 147004, India, also with the School of Computer Science, University of Petroleum and Energy Studies, Dehradun 248007, India, and also with the Department of Computer Science and Information Engineering, Asia University, Taichung 413, Taiwan (e-mail: neeraj.kumar@thapar.edu).

Digital Object Identifier 10.1109/TITS.2021.3095161

network archetype to enable the vehicular communication network with low maintenance, auto-configurable, and global coverage features [6]. Following this, Sadio *et al.* [7] proposed an architecture that is comprised of four layers: Cloud, SDN controller, Fog, and vehicles to offer programmable control, scalability, flexibility, and global knowledge. Moreover, authors used a SDN controller for routing the packets from the server to VANET vice-versa. In recent, SDN enabled control system has also been proposed for efficient traffic management by considering emergency situations in the cities [8], [9]. The above study is evident for the significance of the software-defined control architecture and future dependency of the transportation system on it. Despite IoT and information, & communication technology-enabled transport infrastructure, existing DTLCSs have certain limitations as follow: 1) Exiting DTLCSs are based on centralized or decentralized architectures; centralized controllers are poor in scalability and global coverage due to single point of failure, and distributed controllers are having management and maintenance costs due to lack of programmable control and self-configurable features. 2) Most of the DTLCSs take the decision to route the traffic based on local traffic data and ignore the traffic load balancing around the city which causes traffic jams in few regions of city.

To overcome the limitations, an intelligent traffic-light scheduling framework is proposed, and the significant contributions of the framework, i.e., SDDRL are itemized as follow:

- SDDRL utilizes a software-defined networking architecture to prevent the congestion and smoother the traffic flow in congested areas. In this architecture, each traffic light controller is considered as a control node. RFIDs/camera sensors used to capture the data are considered as normal/common nodes and traffic Cloud/Fog acts as a control server.
- To estimate the effective load on the road lanes, each heterogeneous vehicle is assigned a unique weight. Based on this, DRL model trains its agent which acts as an experienced and intelligent traffic light controller to make effective traffic light control decisions.
- To establish a software-defined control network architecture, parallel agents are deployed on each traffic light controller installed at the respective intersection.
- To prevent the congestion in the network, an algorithm is deployed at the control server which takes input from the real-time vehicular dynamics, and it generates a new phase duration in which DRL agent executes when a certain threshold value is crossed.
- To evaluate the performance of SDDRL, a realistic simulation is developed using SUMO [10], an open-source tool on Indian city OpenStreetMap. The results obtained are compared with several state-of- the-art methods which proved the effectiveness of SDDRL.

II. Related Work

Mir and Hasan [11] proposed a traffic light control system embedded with fuzzy controller and neural network dedicated to minimize queue length and waiting time for car vehicles. Further, Younes and Boukerche [12] proposed a heuristic based solution to tackle emergency vehicles, and model was tested via SUMO simulation, though, priority vehicles were ignored. Abadi *et al.* [13] presented an approach to predict traffic flow in vehicular networks but on limited road traffic data using Monte Carlo simulation technique and auto-regressive model. A fuzzy neural approach was also proposed by Quek *et al.* [14] to predict and analyse road traffic using pseudo outer-product fuzzy neural network with truth-value-restriction, it missed to deal with dense traffic flow. Later, Belletti *et al.* [15] explored a multi task DRL model for expert level control of ramp metering and proved the effectiveness of neural network based deep reinforcement learning for the parameter-known problems. In recent, domain experts [16] recommended the use of Convolutional Neural Network (CNN) with RL technique to control the dynamic traffic light, but this model was tested only on homogeneous vehicles. Following this, N. kumar *et al.* [5] also developed Fuzzy inference based controller and utilized DRL model to act the traffic light in three modes: fair, emergency and priority by accounting heterogeneity among the vehicles. This controller minimizes waiting time and queue length locally and ignored the congestion in the city.

Considering dense structure of traffic light junctions and Vehicular dynamics, software defined control management explored as a network archetype in ITS [17]. Alipio *et al.* [18] also utilized software defined testbed for value based utility implementation and QoS provisioning to manage traffic effectively. With this demand of software defined control network in ITS, SDN based control system was also proposed for an efficient traffic control and management considering emergency situations in the cities [8]. However, authors ignored congestion made at the road lanes due to common traffic. In recent, Balta and Özçelik [17] proposed a model for traffic signal optimization in urban city utilizing SDN based VANET and a 3-stage fuzzy-decision tree, however, authors ignored the heterogeneity among the traffic and load balancing among the dense areas of the city. Van der Pol and Oliehoek [19] proposed coordinated deep reinforcement learners with a new reward considering the penalty function of waiting time, delay, crashes and/or jams and deceleration. However, authors ignored the priority among the vehicles while training the agent. Despite this, there is no real-time monitoring for the congestion and no explicit congestion control signals. In contrast, SDDRL deals with vehicles priority and utilizes a software defined control architecture based coordination to regularly monitor the congestion and generate the congestion prevention signals to handle it. Further, software defined control architecture is enabled by deploying parallel DRL agents to tackle the congestion and smoother the traffic flow in the city.

III. Problem Statement and Solution Approach

This work identified a problem of imbalanced traffic load on the road lanes and tackled by proposing an intelligent traffic light scheduling framework, named as SDDRL. The explored problem is complex due to heterogeneity among vehicles, real-time communication, dense network of traffic light junctions, uncertain vehicular dynamics, and frequent unexpected

congestion. A pictorial representation of the architecture is shown in Fig. 1, it acts in two step; first, it collects vehicular traffic information through the vehicular network [20] with the help of RFID/camera sensors etc., the same is represented in Fig. 1. A trained DRL agent is deployed on Raspberry Pi to process the inputs received through sensors in order to get vehicles traffic state and other parameters, i.e., waiting time, queue length, vehicle speed etc., as considered in many previous studies [16]. With this input data, in step 1, SDDRL makes decisions about the traffic signal phase and its duration. Decision for the traffic light signal phase is taken by the DRL agent that is trained considering heterogeneity among the vehicles, queue length, waiting time and other vehicular dynamics. All traffic light junctions are equipped with a trained DRL agent, and these agents can act in parallel on multiple intersections. In second step, a software defined control enabled vehicular network is created to regularly interact with the agents in order to monitor the congestion. If any congested route is detected, then it generates the congestion prevention signal to minimize inflow and maximize outflow for the respective intersection, otherwise, traffic flows by obeying the signal phase generated by the agent. CS is considered to be deployed on traffic Cloud as shown in Fig. 1.

In general, traffic light controller generates three signals namely green, red and yellow. One traffic light is not sufficient to control the vehicles when inflow of the vehicles is from multiple directions. Therefore, multiple traffic lights work in synchronization at each intersection to manage the traffic flow from the multiple directions. At such traffic light junctions, the traffic lights keep changing in such a way that there is non-conflicting flow of the traffic in multiple directions. Phase duration represents the time interval of staying at one particular signal i.e., green/red/yellow. All phases change cyclically at each junction, and it is called a cycle which keeps on repeating. If any intersection encounters the red traffic signal, no vehicle is allowed to pass through it. On the contrary, if the traffic signal is green, vehicle can have free passage. The goal of this work is to detect relatively high congested regions in the urban city and to minimize the waiting time, queue length and maximize the throughput and speed of the traffic flow in congested areas considering heterogeneity among the vehicles.

IV. Background Study

A. Reinforcement Learning

RL is a machine learning approach in which an agent preforms some actions in an environment with the aim of increase in the aggregated reward. A Markov Decision Process (MDP) is used to describe an environment for RL where the environment is fully observable. Mathematically, MDP is a control process which provides a framework to represent a complex decision making process in a non-deterministic environment. It depends on the Markov property which implies that the future state does not depend on the past state, and it only depends on the present state. In general, it is signified by a 3-tuple (S, A, R) where, S denotes the state space as a set of all possible states for an agent, A denotes the action space as a set of all possible actions available for the agent, and reward

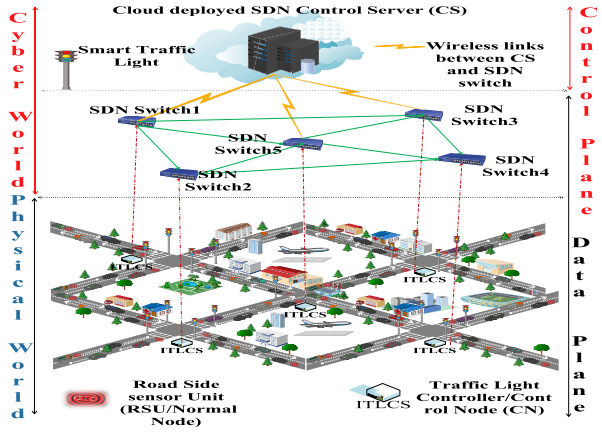


Fig. 1. Cyber-Physical world for software defined control enabled ITS.

R(s, a) defines the reward collected by taking an action a at state s, where, $s \in S$ and $a \in A$. Our objective is to maximize the total rewards of a policy. For example, a reward can be the added score in a game. The domain of R can be $[-\infty, \infty]$. Mathematically, it can be written that a state s_t (at time stamp t) has Markov's property, if and only if;

$$P[s_{t+1}|s_t] = P[s_{t+1}|s_1, s_2, \ldots\ldots, s_t] \tag{1}$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma_k R_{t+k+1} \tag{2}$$

where, G_t returns the total future reward collected by MDP agent from the environment with the discount factor γ. The goal of the MDP is to generate an optimal policy for the decision maker. A policy $(\pi(a|s))$ defines how an agent acts from a specific state. It is the probability of choosing an action a in state s. Mathematically, a policy is defined as follows:

$$\pi(a|s) = P[A_t = a|S_t = s] \tag{3}$$

In RL, interaction between the agent and environment is done in discrete time steps. At each time step, an observation is sent to the agent as a reward. The agent chooses an action from the action state space. This leads to a transition in the state of the environment and change in the reward. The aim of agent is to maximize the total reward.

B. Software Defined Networking in Vehicular Network

SDN is a dynamically manageable, flexible and self-configurable network architecture which is adaptable with Cloud/Fog technologies [21]. SDN decouples the network into two planes i.e., data plane and control plane [21]; data plane is responsible for collecting the data packets from data generating sources and forwarding them to the control plane, and control plane is an intelligent component which is responsible to collect and forward the aggregated data received from data plane to the control server. Control server makes the decisions based on the received data and transfers its signal to the destination nodes at the data plane. Software defined control architecture in vehicular network decouples

the vehicular network into data and control planes [22]. The sensor enabled smart devices (RSUs, cameras etc.) in traffic environment are considered in data plane which are responsible to collect traffic data from the environment and send it to the control plane nodes. The intelligent traffic light controllers are considered as the control plane nodes which implement the aggregating and forwarding functions to send the traffic data to the control server deployed on traffic cloud. Control Server enables SDN based network control switches to act with programmable control, and the underlying infrastructure can be abstracted based on requirements of traffic applications and networking services [17].

V. SDDRL Traffic-Light Scheduling Framework

A. SDDRL Logical Flow

Fig. 2 presents a brief overview of the proposed SDDRL framework. As shown in Fig. 2, SDDRL is composed of two major components: 1) DRL, and 2) SDN. DRL component begins by taking real-time traffic vehicular dynamics as the inputs via sensors deployed at each intersection. RTPE module extracts useful parameters such as Average Waiting Time (AWT), Average Speed (AS), Average Queue Length (AQL) and Throughput (TP) [5], and sends them to NR calculator module which calculates the normalized reward. At the same time, these parameters are forwarded to the Message send/receive module of the SDN component which exchanges the massage between SDN and DRL components. Based on traffic parameters, current traffic state is calculated by RTPE module and sent to the DRL based agent.

At the same time, SDN component activates one of its modules, i.e., $ThresholdCalculator()$ to calculate the threshold (X). If X is less than the pre-decided threshold (th), it generates no change signal as congestion is not detected on the route, else it calls $CongestionPreventionSignalGenerator()$ module to generate congestion prevention signal. This signal is utilized to change/update the phase duration of the respective traffic light controller. Following this phase duration, DRL agent generates the traffic light control signal as the output.

B. The Deep Q-Learning Model

This section details about Q-learning technique used to train the DRL agent that generates traffic light control signals. DRL agent takes positions and velocities of the vehicles in position and velocity matrices as the inputs that are fed to train the model [2]. It outputs a 2×1 vector that contains Q-value corresponding to two possible action, i.e., 0 or 1. Agent chooses an action which has greater Q-value. If the output is 0 then the traffic signal phase remains unchanged else if the output is 1 then the traffic signal phase changes, i.e., red to green or vice versa. Basic principle of Deep Q-learning is to extend traditional Q-learning to continuous state spaces. In MDP based problems, states must be in discrete form which are to be stored in the Q-function. Since, states at a junction are enormous; therefore, traditional method is not suitable to estimate Q-value efficiently. Thus, an approximation of Q-function using a deep learning neural network is warranted.

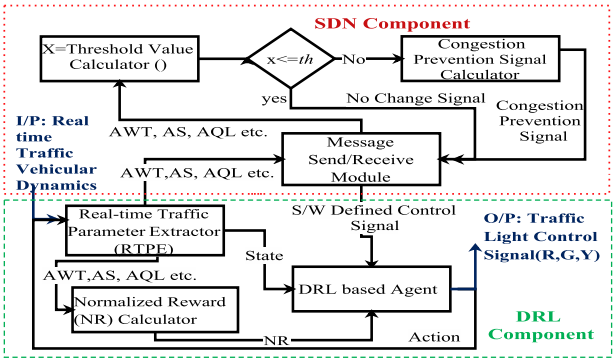
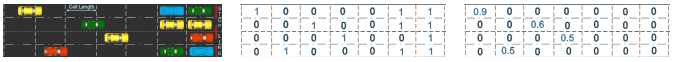


Fig. 2. Logical flow of SDDRL.



(a) Vehicular traffic on the road (b) Corresponding Position matrix (c) Corresponding velocity matrix

Fig. 3. A demonstration of traffic state conversion into input matrix.

Usually, Q-learning uses an online learning approach, in each simulation step, a reward is received and instantly used to improve the learner. Based on the idea of *experience replay*, in each step, past experiences are stored in memory, and a random batch is retrieved to train the neural network to prevent over fitting. The Q-Value is calculated using Bellman equation as [2]:

$$Q^{\pi}(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | s_t, a_t] \quad (4)$$

At each step, Q-value is updated after observing the outcome and reward using following equation:

$$Q(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma \max Q(s', a') - Q(s,a)] \quad (5)$$

where, Q-learning estimates the state-action value function(Q (s,a)) for a target policy that deterministically selects an action of highest value. The RL model is modelled as MDP which is defined as a three-tuple (S, A, R). Three-tuples are defined for the proposed DRL component as follows.

State Space (S): States of the vehicles are captured using two variables, i.e., position (P) and velocity (V) at the intersections. All the lanes joining at the traffic junction are interpreted as square grids with the cell length c considering that no two vehicles fall in the same cell by carefully choosing the value of c [2] as shown in Fig. 3. Each cell contains a two-tuple to store the position and velocity of the vehicles present in that cell. The position value is 0 if no vehicle is present, and 1 if vehicle is present in the cell. The velocity variable (V) stores the normalized speed of the vehicle present in the respective cell. The state is represented as: S = [P,V] where, $P = B^{12\times12} B \in \{0, 1\}$ and $V = Z^{12\times12} Z \in [0, 1]$.

Action Space (A): At a junction, an agent can perform two actions for smoother flow of the traffic based on the observed state. In this model, an agent changes the traffic signal phase (red to green or vice-versa) if the value is 1, or it runs in the same phase if the value is 0, where, $A \in \{0, 1\}$.

Reward (R): In RL, an agent learns from the rewards, it receives a reward after performing some actions; it receives high reward for performing significant action and less in the case of performing less significant action. The reward received

in each episode can either be positive, negative or zero. The agent tries to decrease the waiting time and increase speed for all the vehicles present at the junction. It also tries to minimize the average queue length for each lane. Therefore, the agent calculates reward based on two observations: first, before the action is performed, and second after the action is performed. Let, r_1 and r_2 are the rewards before and after action performed respectively. The domain of R is $[-\infty, \infty]$. Considering this scenario, reward is calculated as:

$$reward = r_2 - r_1 \tag{6}$$

$$r_1 = \sum_{j=1}^{x} W_j * (NWT_j - NS_j) + \sum_{k=1}^{nL} W_k(ND_k + NQL_k) \tag{7}$$

$$r_2 = \sum_{j=1}^{y} W_j * (NWT_j - NS_j) + \sum_{k=1}^{nL} W_k(ND_k + NQL_k) \tag{8}$$

$$Xvar = \frac{Xvar_{curr} - Xvar_{min}}{Xvar_{max} - Xvar_{min}} \tag{9}$$

where, nL is the number of lanes at the junction, and x and y are the number of vehicles at the intersection before and after the execution of an action respectively. W_j is the weight of $j-th$ vehicle, and NWT_j and NS_j are the normalized waiting time and speed of $j-th$ vehicle respectively. ND_k, W_k and NQL_k are the normalized delay, vehicles load and queue length on $k-th$ lane respectively. The reward with positive value specifies that chosen action is better performed for the current state of vehicles at the intersection. $Xvar$ is the normalized variable for the respective parameter, $Xvar_{curr}$, $Xvar_{min}$ and $Xvar_{max}$ represent current, minimum and maximum value of the normalizing traffic parameter i.e., $Xvar$.

C. Convolutional Neural Network

The layered architecture for the proposed CNN is shown in Fig. 4. The position and velocity matrices of size 12×12 are fed as the inputs. First convolution layer consists 16 filters of size 4×4 with a stride value 2, and the leaky Rectified Linear Unit (ReLU) is used as an activation function, and equation for the same is given as follows:

$$f(x) = \begin{cases} x & x \geq 0 \\ \beta x & x < 0 \end{cases} \tag{10}$$

Second layer consists 32 filters of size 2×2 with a stride value 1 followed by leaky ReLU activation function. Third layer is concatenation layer, the output from position and velocity matrices are firstly flatten, then both are concatenated along with the previous state output. Next layer is fully connected linear layer of size 128. This layer is followed by yet another fully connected layer of the size 64. The last layer is the output layer consisting of two fully connected linear units which gives the Q-value corresponding to both

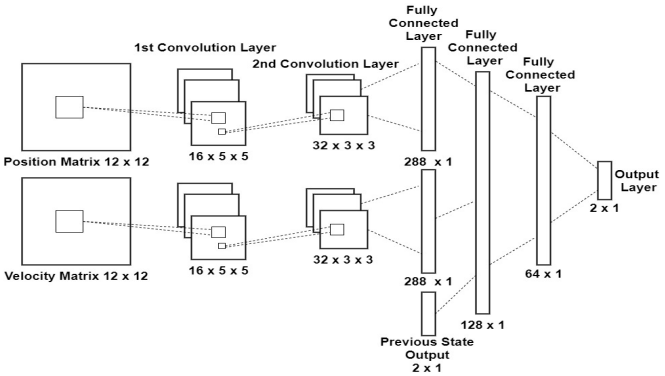


Fig. 4. The proposed convolution neural network architecture.

possible actions, an agent can perform. The weight parameters of the CNN are initialized using random weights. At the starting of every episode, an agent observes the simulated traffic environment to extract the position and velocity matrices to define the state space S_t of vehicles at the junction. By processing these inputs, an agent performs the action A_t with the maximum Q-value. After executing the action A_t, an agent receives the reward R_t according to the likeliness of the performance. As a result of chosen action, an agent gets a new state S_{t+1}. The 4-tuple (S_t, A_t, R_t, S_{t+1}) is stored in experience replay memory as a training example for the training of CNN. Due to limited memory, after a memory threshold limit, old experiences are replaced by the new ones. This way, CNN is trained using the experiences (S_t, A_t) stored in the experience replay memory. The network learns this mapping from state space to action space by minimizing the mean squared error(J(θ)). The equation for the same is given as follows:

$$J(\theta) = \frac{1}{M}\sum_{t=1}^{M}[(R_t + \alpha max_{a'}Q(s_{t+1}, a', \theta) - Q(s_t, a_t, \theta)]^2 \tag{11}$$

where, M is the size of training dataset. The value of weight parameter (θ) is updated using a variation of stochastic gradient descent called RMSProp algorithm.

D. Software Defined Network Component

SDN component is responsible to prevent the congestion and to balance the traffic in the city. To do this, SDN control server receives real-time traffic parameters periodically from all the junctions, and then expected congestion value i.e., X_i for i^{th} junction is calculated as follow.

$$X_i = \phi NAWT_i + \chi NAQL_i - \psi AS_i \tag{12}$$

where, ϕ, χ and ψ are waiting time, queue length and speed factors respectively, and $NAWT_i$, $NAQL_i$ and NAS_i are the normalized value of average waiting time, average queue length and average speed respectively (normalized using Eq. 9) for the i_{th} junction. If $X_i < th$ then SDN controller sends a "no change signal" to the agent installed at junction, otherwise, congestion prevention signal is generated and sent to the agent to manage the traffic flow accordingly. Congestion prevention signal contains the new phase duration for the respective agent.

Algorithm 1 SDDRL

Input: Number of junctions (N), Position Matrix (P), Velocity Matrix (V), Numbers of Phases at i_{th} Junction (NP_i), Average Waiting Time(AWT), Average Queue Length (AQL) and Average Speed (AS)
Output: Traffic light control signal (red/green/yello).
Notation:
A_i: Action performed by i^{th} agent at the respective junction
T_i: Phase Duration for i_{th} junction
W_i: Weight Matrix for i_{th} junction

for each $i \in N$ **do in parallel**
  TrainDRLAgent(P, V);
  Deploy DRL based trained agent at i^{th} junction;
  Initialize: $phase_i \leftarrow 0$
  Initialize: $Step_i \leftarrow 0$
  $W_i \leftarrow getWeightMatrix(P)$;
  while $Step_i < SimulationSteps$ **do**
    if $Step_i == 0$ **then**
      $T_i = getInitialPhaseDuration(W_i)$
    end if
    $A_i \leftarrow agent.getAction(P, V)$;
    if $A_i == 1$ **then**
      $phase_i = (phase_i + 1) \mod NP_i$;
      Change Traffic Signal Phase;
      SetPhaseDuration(T_i);
    end if
    if $A_i == 0$ **then**
      Do not Change Traffic Signal Phase;
      SetPhaseDuration(T_i);
    end if
    $Step_i = Step_i + 1$;
    if $Step_i \mod 10 == 0$ **then** ◁ Send local traffic information to CS in every 10s.
      SendMsgToCS(AWT_i, AQL_i, AS_i);
      T_i = SDC (N, AWT_i, AQL_i, AS_i);
      ReceiveMsgFromCS(T_i);
    end if
  end while
end for

Algorithm 2 TrainDRLAgent

Input : P, V, Replay Memory Size (M), Exploration Rate (ϵ), Learning Rate (α), Discount Factor (γ), Number of Epochs (E).
Notation:
θ: Weight parameter in deep neural network
S: State tuple consists of position matrix and velocity matrix
A: Action performed by agent
R: Reward for performed action
m: Replay Memory

for each $e \in E$ **do**
  Initialize traci environment as Env;
  for $S \leftarrow getCurrentState(TlsID)$; **do**
    $steps \in SimulationSteps$
    $A \leftarrow agent.getAction(S)$;
    $R \leftarrow getReward(W, S)$;
    $S' \leftarrow getNewState(TlsID)$;
    if $|m| < M$ **then**
      Remove oldest data from replay memory;
    end if
    Append $\langle S, A, R, S' \rangle$ in m;
    $J = \frac{1}{M} \sum_{t=1}^{M} [(R_t + \alpha max_{A'} Q(S_{t+1}, A', \theta) - Q(s_t, a_t, \theta)]^2$;
    Update θ with J using RMSProp algorithm;
    Update the value of ϵ
  end for
end for

SDN component contains a message send/receive module which is responsible for all message passing between the data plane and the control plane. The data plane sends average of waiting time, queue length and speed to the control plane. As per the study [23], [24], researchers suggested average speed, delay and queue length are the key parameters to quantify the congestion. Therefore, all three parameters are used to calculate X.

E. Algorithm Description

Algorithm 1: SDDRL begins by taking the inputs as shown in Algorithm 1. Based on the inputs, DRL agent is trained by calling $TrainDRLAgent(P, V)$ and deployed at the respective traffic light junction. The $phase_i$ and $Step_i$ are initialized to 0, where $0 \leq i \leq N - 1$. Before going to simulation steps, W_i is calculated, to do this, vehicle position value is multiplied with the weight assigned for the respective vehicle based on its priority. For the priority weights, $Police = 3$, $Bus = 2$, $car = 1$, $Truck = 1$, $Ambulance = 5$, $Fire\_brigade = 5$ are fixed. However, these weights can be changed as per the application's priority requirements. Next, SDDRL enters in its execution loop for the number of simulation steps. For the first step, phase duration (T_i) is calculated using $getInitialPhaseDuration(W_i)$. Next, $agent.getAction(P, V)$ is called which generates an action (0 or 1) that is stored in A_i, if $A_i == 1$ then traffic light phase is increased by one and changed by setting phase duration T_i using $SetPhaseDuration(Ti)$. Otherwise, phase remains same, and T_i is set for the current phase. In this scenario, minimum phase duration is assumed to be 10 seconds, therefore, after every 10 seconds, traffic parameters are sent to CS using $SendMsgToCS(AWT_i, AQL_i, AS_i)$, based on these parameters, Software Defined Control (SDC) module calculates new T_i that is received by i^{th} agent. Same procedure is repeated for the number of simulation steps.

Algorithm 2: $TrainDRLAgent$ function is used to train the DRL agent. It takes P, V, hyperparameters such as M, ϵ, α, γ and E as the inputs. The simulation environment is initialized using sumo-traci. The $getCurrentState(TlsID)$ stores the current state of the junction as S that is a tuple consisting of position matrix and velocity matrix, where $TlsID$ represents the list of traffic light junctions IDs in the network.

Algorithm 3 getIntialPhaseDuration

Input : W.
Output: Returns Initial Phase Duration
1: *Intialize:* $Total\_Sum = 0, Sum[4] = \{0, 0, 0, 0\}, T$;
2: **for each** $j \in \{1, 4\}$ **do in parallel**
3: | **for** v: *vehicles in* j^{th} *direction* **do**
4: | | $Sum[j] = Sum[j] + W[v]$;
5: | **end for**
6: | $Total_Sum = Total\_Sum + Sum[j]$;
7: **end for**
8: **for each** $j \in \{1, 4\}$ **do in parallel**
9: | *return* $\frac{Sum[j]}{Total\_Sum} \times T$;
10: **end for**

At each simulation, the parameters of convolutional neural network are updated. The $agent.getAgent(s)$ outputs the action taken by the agent based on the current state of traffic. The $getReward(W, S)$ generates a reward for the performed action based on the current state(S) and corresponding weight matrix(W) of the vehicles. The $getNewState(TlsID)$ gives a new state after an action is performed. The 4-tuples $< S, A, R, S' >$ are stored in the replay memory. Now, cost function(J) is calculated and used to update parameters of CNN using RMSProp algorithm.

Algorithm 3: The $getIntialPhaseDuration()$ is used to generate initial phase duration for the agent based controller which takes the input W. For a 4-lane junction, $Total\_Sum$ and Sum [4] $= \{0, 0, 0, 0\}$ are initialized, and for the tri-junction, $Sum[3]$ is used. Phase cycle time duration (T) is fixed to 60 seconds. For $j-th$ lane of the junction, sum of all vehicles weight is calculated and stored in $Sum[j]$. Following this, $Total\_Sum$ is calculated, and T is distributed for all the phases proportional to the sum of vehicle weights.

Algorithm 4: SDC executes when it receives the traffic parameters *i.e.*, AWT_i, AQL_i, AS_i from i^{th} agent. In lines 1 to 5, some constants are initialized, however, corresponding values can be changed as per the application requirements. For the i^{th} agent, $NAWT_i$, $NAQL_i$ and NAS_i are calculated, based on these, expected congestion value (X_i) is obtained. If $X_i \geq th$ then T_i is increased by 20%, and T_j for its j^{th} neighbouring agent is decreased by 20%, where, $0 \leq j \leq |Neighbours| - 1$. Finally, respective T_i is sent to i^{th} agent in which it executes.

The time-complexity of Algorithm 1 is $O((SimulationStep \times epocs) + SimulationSteps + MaxQueuelength)$, and the Algorithm 2 is to train an agent which runs in $O(SimulationSteps \times epocs)$. The Algorithms 3 and 4 run in $O(MaxQueuelength)$, and $O(N)$ respectively.

VI. Performance Evaluation

The performance analysis of the proposed SDDRL is done through the simulation prepared on OpenStreetMap of an Indian city, utilizing a well-known open source simulator, i.e., SUMO [10]. The effectiveness of the SDDRL is measured on several performance metrics: AWT, AQL, TP and AS.

Algorithm 4 SoftwareDefinedControl(SDC)

Input : N, AWT_i, AQL_i, AS_i
Output: Updated phase duration for each Junction T_i.
1: $th \leftarrow 0.6$; ◁ Expected congestion threshold value
2: $\phi \leftarrow 0.4$; ◁ Waiting time factor
3: $\chi \leftarrow 0.3$; ◁ Queue length factor
4: $\psi \leftarrow 0.3$; ◁ Speed factor
5: $t \leftarrow 10$;
6: **for** $i \in N$ **do**
7: | *ReceiveMsgFromAgent*(AWT_i, AQL_i, AS_i);
8: **end for**
9: **for each** $i \in N$ **do in parallel**
10: | $NAWT_i = \frac{AWT_i - AWT_{min}}{AWT_{max} - AWT_{min}}$;
11: | $NAQL_i = \frac{AQL_i - AQL_{min}}{AQL_{max} - AQL_{min}}$; $NAS_i = \frac{AS_i - AS_{min}}{AWT_{max} - AS_{min}}$;
13: | $X_i = \phi NAWT_i + \chi NAQL_i - \psi NAS_i$; **if** $th \leq X_i$ **then**
15: | | $T_i = T_i + 0.2t$;
16: | **end if**
17: | **for** $j \in Neighbour_i$ **do**
18: | | $T_j = T_j - 0.2t$;
19: | **end for**
20: | *SendMsgToAgent*(T_i);
21: **end for**

Fig. 5. The map used for SUMO simulation study.

For the purpose of fair comparative study, we implemented several methods using the same simulation environment and testing parameters. Thus, SDDRL is compared with the diverse nature traffic light controllers, i.e., FCTL [25], NFM [26], DQN [2], Max Pressure Algorithm (MPA) [27] and Fix Time Algorithm (FTA) to verify its effectiveness.

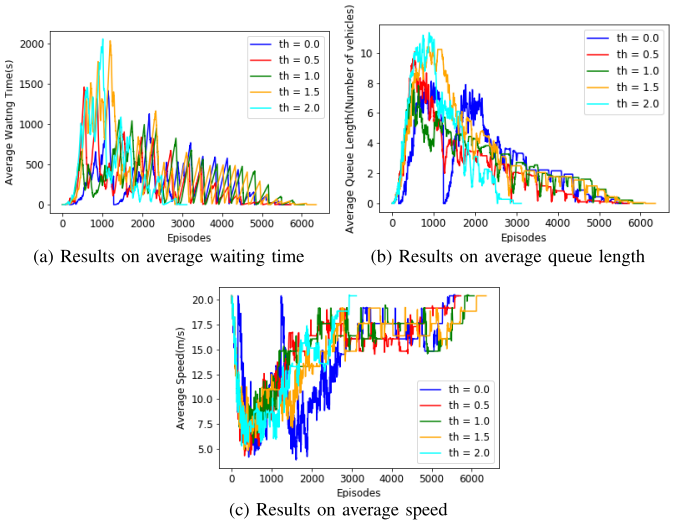
A. Simulation Parameters and Methodology

In order to implement the proposed SDDRL, all modules were coded in Python and, vehicular network simulation was prepared in SUMO. The DRL component of the SDDRL was implemented utilizing Keras that runs on the top of Tensorflow [28]. The OpenStreetMap of Gwalior city, India was downloaded as.osm file as shown in Fig. 5. For the simulation, the same file was converted into road lanes and traffic using SUMO conversion tool, i.e., netconvert. In order to control the traffic behaviour in SUMO simulation via

python implementation, SUMO offers a Traffic Control Interface (TRACI) as a part of sumo module. With the help of TRACI, SUMO and Python integration is quite effective; this enables control of SUMO traffic flow through written algorithmic code in Python. The SUMO simulation parameters are listed in Table I. To generate the traffic on an OpenStreetMap in SUMO simulation, randTrips.py script is utilized. In this script, the arrival rate is controlled by option –period <FLOAT> (default 1). By default, this generates vehicles with a constant period and arrival rate of, i.e., 1/period per second. With the value less than 1, multiple vehicles arrival per second can be achieved, and this arrival rate is randomized using a binomial distribution. As per the real traffic scenario of the Gwalior city during the rush-hours, arrival rate is observed between [0.4, 0.9] by the team via a continuous manual observation of seven days. Therefore, the probability in the randTrips.py script was set accordingly. Following this, the heterogeneous traffic density varies according to the traffic conditions during the run. The duration of yellow signal is a constant of 3 seconds; that is considered as a transition phase for the signal change,i.e., green to red or vice versa. Deep network is simulated using replay memory size ($M = 3000$), exploration rate ($\epsilon = 0.1$), minimum batch size ($B = 32$), learning rate ($\alpha = 0.0002$), discount factor ($\gamma = 0.95$), and leaky ReLU ($\beta = 0.01$). Software defined control is simulated using 5 control nodes which are connected to a control server, waiting time factor ($\phi = 0.4$), queue length factor ($\chi = 0.3$), speed factor ($\psi = 0.3$) and $th = 0.6$. As the minimum and maximum values of the proposed function, as shown in Eq. 12, can be in the range of $(-1, 2)$. If the sum of $NAWT$ and $NAQL$ is 0 and NAS is at its maximum value i.e., 1, it achieves the lowest valve i.e., -1 as it shows congestion-free traffic flow. It logically implies that if the value of the proposed function, as shown in Eq. 12, is less than zero then there is no congestion during the traffic flow. However, if the value exceeds the zero then the possibility of congestion arises, and it can be maximum 2 considering the case $NAWT+NAQL = 2$ and $NAS = 0$, this signifies the traffic jam as shown in Fig. 6. Based on the sensitivity analysis in Fig. 6, it is observed that the traffic flow is smoother throughout the simulation episodes, and other performance metrics are also better if the threshold congestion value is set to 0.5 in comparison to the other scenarios. If it goes higher than 0.5 then results are getting worse. Thus, the suggested threshold value is 0.6. However, this value can be adjusted as per the requirement of the application.

TABLE I
SUMO SIMULATION TRAFFIC PARAMETERS

SUMO Simulation Parameters	Value
Length of Car	$4.5m$
Length of Bus/Truck	$7m$
Max. Lane Speed	$20m/s$
Max. Speed of Car	$15m/s$
Max. Speed of Bus/Truck	$10m/s$
Other Vehicle's Max Speed	$13.9m/s$
Max. Acceleration of Car	$2.5m/s^2$
Max. Acceleration of Bus/Truck	0.8 m/sec^2
Max. Acceleration of Other Vehicles	1.8 m/sec^2
Max. Acceleration of other vehicles	1.8 m/sec^2
Max. De-acceleration of all Vehicles	4.0 m/sec^2
Driving Proficiency(σ)	0.5
Detector Frequency	60
Min. Gap between Two Vehicles	$2m$
Maximum Number of Lanes	4
Minimum Number of Lanes	2
Road lane type	Unidirectional
Arrival Rate for default Binomial Distribution (λ)	[0.4,0.9]
Number of Traffic Light Junctions	16
Latitude of the map	26.161674
Longitude of the map	78.072584



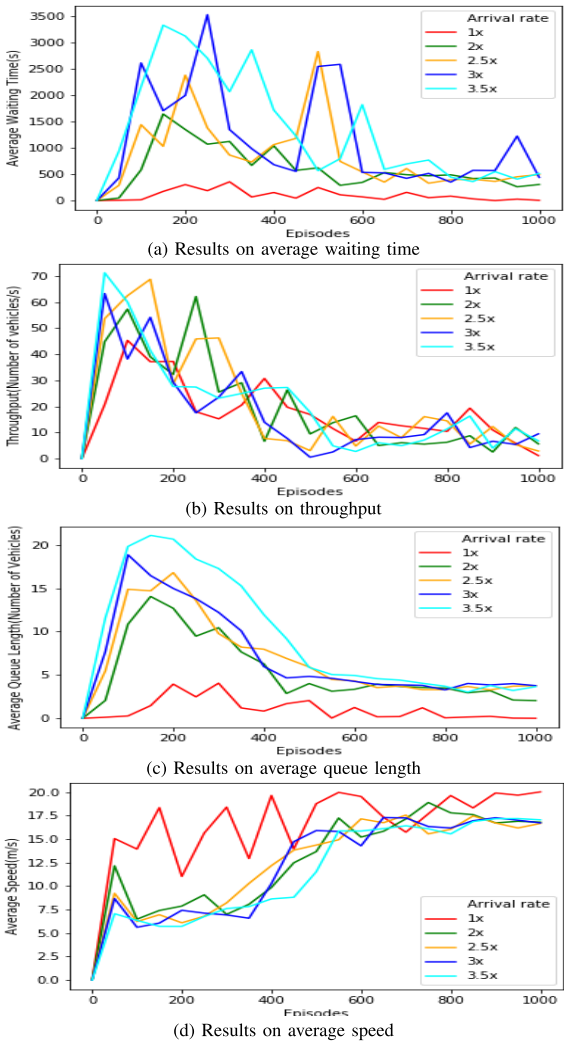
(a) Results on average waiting time (b) Results on average queue length
(c) Results on average speed

Fig. 6. Comparative results of SDDRL on varied congestion threshold values.

B. Results and Discussions

This sub-section provides a descriptive analysis of SDDRL's experimental findings and comparative study with other state of the art algorithms on various performance metrics. To test the performance behaviour of the SDDRL, four major performance metrics i.e., Average Waiting Time (AWT), Throughput (TP), Average Queue Length (AQL), and Average Speed (AS) (for the details see [5]).

1) SDDRL Performance Behavior Study on Varied Arrival Rates of the Vehicles: The behaviour of the SDDRL is analysed for various performance metrics by conducting a set of experiments on varied arrival rates. Fig. 7 (a), 7(b), 7(c) and 7(d) show the results corresponding to AWT, TP, AQL and AS performance metric respectively, where X-axis corresponds to episodes, and the Y-axis shows the performance metric. In Fig. 7, 1x, 2x, 2.5x, 3x and 3.5x represent the increasing vehicles arrival rate. From the Fig. 7, it is observed that AWT and AQL are minimal for the very low arrival rate (x) which is obvious, however, SDDRL maintains the performance for the higher arrival rates (2.5x, 3x, and 3.5x) or congested routes that is the main objective of this work. In contrast, a reciprocal behaviour of SDDRL is observed for AS that is desired. Further, the TP is significantly maintained for all the scenarios. However, it is observed that TP is decreasing by increasing the number of episodes. As the whole simulation is divided into episodes. At the end of the simulation i.e., after a significant number of episodes passed, the numbers of vehicles



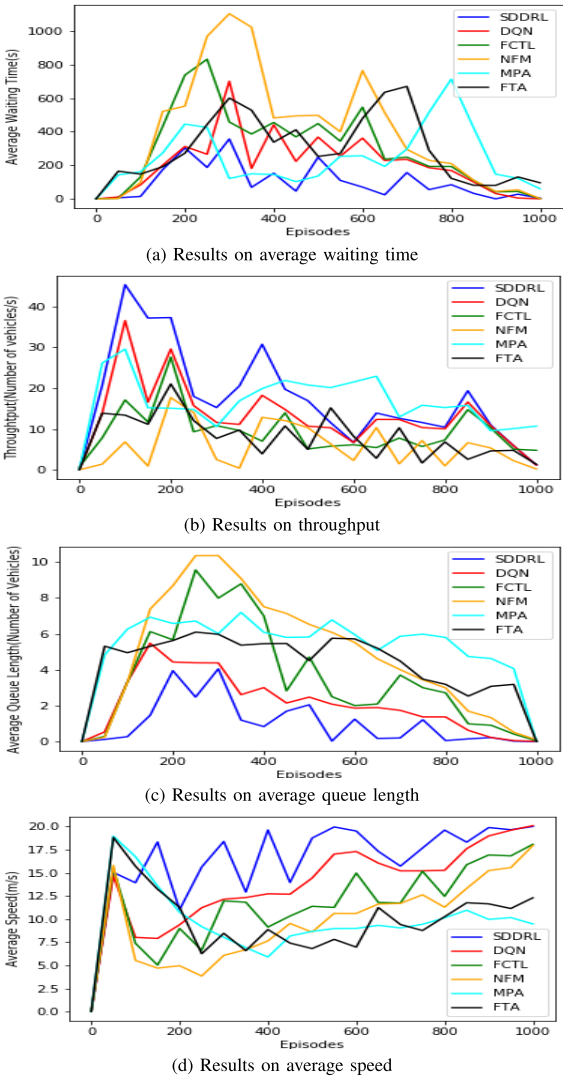
(a) Results on average waiting time

(b) Results on throughput

(c) Results on average queue length

(d) Results on average speed

Fig. 7. Comparative results of SDDRL on varied arrival rates of the vehicles.



(a) Results on average waiting time

(b) Results on throughput

(c) Results on average queue length

(d) Results on average speed

Fig. 8. Comparative results of SDDRL with other state of the art algorithms.

decrease in the simulation as the vehicles completed their respective journey, thus, TP is decreasing when number of episodes have increased. In brief, results indicate that despite of increasing the arrival rates, the minor changes in the performance behaviour of the SDDRL have been recorded. Hence, results in Fig. 7 verify the effectiveness of the SDDRL during the peak hours of traffic. The reason for the same is that the DRL component of the proposed SDDRL is trained by an effective reward considering several traffic dynamics, and SDN control server regularly monitors the traffic congestion and takes the actions in real-time accordingly.

2) Comparative Study: In order to analyse the comparative performance behaviour, the proposed SDDRL is compared with the diverse nature of state of the art models such as deep re-enforcement learning based model: DQN [2], Fuzzy Inference based DTLCSs: FTLC [25] and NFM [26], max-pressure based DTLCS: MPA [27] and fixed-time based DTLCS: FTA. Fig. 8(a), 8(b), 8(c) and 8(d) show the comparative results corresponding to AWT, TP, AQL and AS performance metrics respectively. From the Fig. 8(a), it is analysed that SDDRL improves its performance 66.62%, 49.70%, 30.3%, 54.77% and 28.34% over NFM, FCTL, MPA, FTA and DQN respectively for the AWT metric. Similarly, Fig. 8(b) shows the results corresponding to TP performance metric, and results indicates that SDDRL improves 66.60%, 47.06%, 26.88%, 54.41% and 24.76% performance over NFM, FCTL, MPA, FTA and DQN respectively. Further, SDDRL also improves its AQL performance 69.80%, 59.25%, 61.73%, 63.38% and 30.89% over NFM, FCTL, MPA, FTA and DQN respectively as shown in Fig. 8(c). Finally, SDDRL also beats NFM, FCTL, MPA, FTA and DQN with 40.90%, 29.55%, 42.83%, 43.67% and 16.62% respectively in terms of AS performance improvement as shown in Fig. 8(d). From the results, it is observed that SDDRL have achieved substantial improvement in AWT, TP, AQL and AS in comparison to several state of the art models. As SDDRL takes the real-time traffic parameters as the inputs and agent is trained based on reward which utilizes multiple traffic parameters to produce an effective reward, due to this, an agent generates the traffic light control signals much closer to the traffic conditions considering heterogeneous traffic load presented at each lane.

TABLE II
COMPARATIVE RESULTS OF SDDRL WITH VARYING NUMBER OF JUNCTIONS

Model	# of Junctions	AQL	AS	AWT	TP
SDDRL	4	2.81	15.24	68.53	28.51
	8	3.31	14.58	164.41	27.27
	12	3.31	14.52	164.41	25.72
	16	4.28	12.96	234.75	25.23
DQN	4	5.65	12.43	72.56	24.6
	8	15.32	10.25	342.53	23.65
	12	18.78	6.43	678.55	20.34
	16	21.46	4.56	1245.67	19.32
MPA	4	7.34	8.65	78.54	23.24
	8	17.65	5.87	467.67	21.45
	12	20.76	4.60	1028.76	20.55
	16	23.56	3.21	1456.43	17.76
FCTL	4	11.68	10.02	79.64	22.24
	8	15.12	8.72	2232.78	22.07
	12	18.43	8.85	4700.91	20.32
	16	50.30	3.30	17288.83	18.33

Further, SDN controller regularly monitors congested routes and generates new and effective phase duration based on expected congestion scenarios to prevent the congestion at the intersections. Following this phase duration, an agent generates traffic light control signals to flow the traffic. With this cooperative execution of both components i.e., DRL agent and SDN controller, SDDRL ensures smooth flow of the traffic, and results in significant improvement in AWT, TP, AQL and AS performance metrics. In order to analyze the scalablity of the SDDRL, a set of experiments are done on varied number of junctions, and results are compared with best performing state of the art models as shown in Table II. From the results, it is observed that the SDDRL performs relatively better than other models when the number of junctions are increased. As SDDRL tries to tackle the problem on two fronts i.e., DRL agent is deployed at each junction to generate the traffic light signal and SDN controller which tries to regulate the steady and smooth flow of traffic on in congested regions. Additionally,the priority assignment on vehicles played a significant role in minimizing the AWT for the ambulance and police vehicles. We noted the AWT for the [ambulance, police vehicles] with and without any priority as [0.9, 1.2] and [3.3, 3.3] seconds on respectively. Further, the average computing time of the trained DRL agent to generate a signal phase is observed as 0.0165 seconds which is acceptable to run in real-time scenarios.

VII. CONCLUSION

The proposed SDDRL system takes into account vehicle heterogeneity and road lane congestion when making intelligent traffic light control decisions. The SDDRL majorly composed of two components: 1) DRL based component which takes the real-time traffic vehicular dynamics as the inputs to train an agent based on the proposed reward function, and it generates the traffic light control signals i.e., red/yellow/green in the given phase duration, and 2) SDN component which is dedicated to periodically monitor the dense intersections and generates a phase duration to prevent the congestion, following this, agent acts. To verify the effectiveness of the SDDRL, a realistic simulation is developed utilizing an open source simulator tool Simulation of Urban MObility (SUMO) on Gwalior city of India. SDDRL is verified on several performance metrics on increasing vehicle arrival rates to analyse the behaviour in congested scenarios. Further, results of SDDRL are compared with several Fuzzy, deep re-enforcement learning and max-pressure based state of the art algorithms which proved the effectiveness of SDDRL.

In the future work, this framework will be explored explicitly for emergency service aware traffic light control system and Social Internet of Vehicles (SIoVs) to offer congestion free services in the smart transportation system.

REFERENCES

[1] J. Wan, D. Zhang, S. Zhao, L. T. Yang, and J. Lloret, "Context-aware vehicular cyber-physical systems with cloud support: Architecture, challenges, and solutions," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 106–113, Aug. 2014.

[2] X. Liang, X. Du, G. Wang, and Z. Han, "A deep Q learning network for traffic lights' cycle control in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1243–1253, Jan. 2019.

[3] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Design of reinforcement learning parameters for seamless application of adaptive traffic signal control," *J. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 227–245, Jul. 2014.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[5] N. Kumar, S. S. Rahman, and N. Dhakad, "Fuzzy inference enabled deep reinforcement learning-based traffic light control for intelligent transportation system," *IEEE Trans. Intell. Transp. Syst.*, early access, Apr. 20, 2020, doi: 10.1109/TITS.2020.2984033.

[6] M. Collotta, L. L. Bello, and G. Pau, "A novel approach for dynamic traffic lights management based on wireless sensor networks and multiple fuzzy logic controllers," *Expert Syst. Appl.*, vol. 42, no. 13, pp. 5403–5415, Aug. 2015.

[7] O. Sadio, N. Ibrahima, and C. Lishou, "SDN architecture for intelligent vehicular sensors networks," in *Proc. Int. Conf. Comput. Modelling Simulation*, Mar. 2018, pp. 139–144.

[8] A. Rego, L. Garcia, S. Sendra, and J. Lloret, "Software defined network-based control system for an efficient traffic management for emergency situations in smart cities," *Future Gener. Comput. Syst.*, vol. 88, pp. 243–253, Nov. 2018.

[9] N. Kumar, N. Chilamkurti, and J. H. Park, "ALCA: Agent learning–based clustering algorithm in vehicular ad hoc networks," *Pers. Ubiquitous Comput.*, vol. 17, no. 8, pp. 1683–1692, Dec. 2013.

[10] 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, pp. 128–138, Dec. 2012. [Online]. Available: http://elib.dlr.de/80483/

[11] A. Mir and A. Hasan, "Fuzzy inference rule based neural traffic light controller," in *Proc. IEEE Int. Conf. Mechatronics Automat.*, Aug. 2018, pp. 816–820.

[12] M. B. Younes and A. Boukerche, "An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems," *Wireless Netw.*, vol. 24, no. 7, pp. 2451–2463, Oct. 2018.

[13] A. Abadi, T. Rajabioun, and P. A. Ioannou, "Traffic flow prediction for road transportation networks with limited traffic data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 653–662, Apr. 2015.

[14] C. Quek, M. Pasquier, and B. B. S. Lim, "POP-TRAFFIC: A novel fuzzy neural approach to road traffic analysis and prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 2, pp. 133–146, Jun. 2006.

[15] F. Belletti, D. Haziza, G. Gomes, and A. M. Bayen, "Expert level control of ramp metering based on multi-task deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1198–1207, 2017.

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

[17] M. Balta and İ. Özçelik, "A 3-stage fuzzy-decision tree model for traffic signal optimization in urban city via a SDN based VANET architecture," *Future Gener. Comput. Syst.*, vol. 104, pp. 142–158, Mar. 2020.

[18] M. I. Alipio, A. G. A. Co, M. F. C. Hilario, and C. M. C. Pama, "Value-based utility implementation in software-defined testbed for sensor data traffic management," *Future Gener. Comput. Syst.*, vol. 101, pp. 737–746, Dec. 2019.
[19] E. van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," in *Proc. Learn., Inference Control Multi-Agent Syst. (NIPS)*, 2016.
[20] W. Farooq, M. A. Khan, and S. Rehman, "A cluster based multicast routing protocol for autonomous unmanned military vehicles (AUMVs) communication in VANET," in *Proc. Int. Conf. Comput., Electron. Electr. Eng. (ICE Cube)*, Apr. 2016, pp. 42–48.
[21] N. Kumar and D. P. Vidyarthi, "A green routing algorithm for IoT-enabled software defined wireless sensor network," *IEEE Sensors J.*, vol. 18, no. 22, pp. 9449–9460, Nov. 2018.
[22] A. Boukerche, H. A. B. FOliveira, E. F. Nakamura, and A. A. F. Loureiro, "Vehicular ad hoc networks: A new challenge for localization-based systems," *Comput. Commun.*, vol. 31, no. 12, pp. 2838–2849, Jul. 2008.
[23] Y. Jiang, "Estimation of traffic delays and vehicle queues at freeway work zones," in *Proc. 80th Annu. Meeting Transp. Res. Board*, Washington, DC, USA, 2001.
[24] M. Aftabuzzaman, "Measuring traffic congestion—A critical review," in *Proc. 30th Australas. Transp. Res. Forum*, 2007, pp. 1–16.
[25] M. Kafash, M. B. Menhaj, M. J. Motahari Sharif, and A. Maleki, "Designing fuzzy controller for traffic lights to reduce the length of queues in according to minimize extension of green light time and reduce waiting time," in *Proc. 13th Iranian Conf. Fuzzy Syst. (IFSC)*, Aug. 2013, pp. 1–6.
[26] E. Azimirad, N. Pariz, and M. B. N. Sistani, "A novel fuzzy model and control of single intersection at urban traffic network," *IEEE Syst. J.*, vol. 4, no. 1, pp. 107–111, Mar. 2010.
[27] P. Mercader, W. Uwayid, and J. Haddad, "Max-pressure traffic controller based on travel times: An experimental analysis," *Transp. Res. C, Emerg. Technol.*, vol. 110, pp. 275–290, Jan. 2020.
[28] M. Abadi *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/

Neetesh Kumar (Member, IEEE) received the M.Tech. and Ph.D. degrees from the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi. He has qualified GATE and NET+JRF. He is currently working as a Faculty Member with IIT-Roorkee, India. He has published several SCI index research publications in world's top tier publishers, like IEEE journals and transactions, Elsevier journals (including *Future Generation Computer Systems* (FGCS) and *Information Sciences*), and Springer Journals (*Soft Computing* and *Computing*). Broadly, his research interests include algorithm design, high-performance computing (cloud, fog, and parallel computing), applied evolutionary computing, software-defined networking, WSN, and intelligent transportation systems. He has received several awards, such as the Best Paper Award and the Special Mention Award. He is also acting as a lead PI for several sponsored projects from IIT Roorkee/DST/CSIR agencies, Government of India, and three proposals are in under process by several agencies.

Sarthak Mittal received the integrated master's degree from ABV-IIITM, Gwalior. He is currently working as a Software Developer at Turtlemint. His areas of interest are deep learning, reinforcement learning, data structures, algorithms, and ITS simulations.

Vaibhav Garg received the B.Tech. degree in computer science engineering from ABV-IIITM Gwalior in 2021. His fields of interest are natural language processing, machine learning, and nature inspired computing.

Neeraj Kumar (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from SMVD University, Katra, Jammu and Kashmir, India. He was a Post-Doctoral Research Fellow with Coventry University, Coventry, U.K., where he is currently a Visiting Research Fellow. He is also working as a Full Professor with the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, Punjab. He is also an internationally renowned researcher in the areas of VANET, CPS smart grid, the IoT mobile cloud computing, big data, and cryptography. He has visited many countries mainly for the academic purposes. He has many research collaborations with premier institutions in India and different universities across the globe. He has supervised 15 Ph.D. students and five students are currently pursuing their thesis. He has also supervised more than 25 M.E./M.Tech. theses. He has published more than 400 technical research papers in leading journals and conferences from IEEE, Elsevier, Springer, John Wiley, and Taylor and Francis. He has total research funding from these agencies of more than 60 000 000 under different schemes from GOI. He has H-index of 77 (according to Google scholar July 2021) with 19700 citations to his credit. He has won many prestigious awards from different venues. He is an Editorial Board Member of *IEEE Network*, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, *IEEE Communications Magazine*, the *International Journal of Communication Systems* (Wiley), *Security and Communication Networks* (John Wiley), and *Journal of Network and Computer Applications* (Elsevier).