of managing traffic congestion in different metropolitan areas. To effectively monitor traffic flow, coordination between traffic signals at intersections must be improved; otherwise, congestion will continue to propagate across time to many other neighbouring intersections [2]. Interdependence among traffic signals at various intersections seems to be so significant for a metropolitan area that the variations in traffic signals can have repercussions on each other. Current traffic signal control methods also rely heavily on oversimplified knowledge and regulatory approaches, whereas recently there are enormous data, improved processing resources and sophisticated methods for driving smart transport. With an increase in population and modern innovations in transport systems, transport has developed into smart networks known as ITS [3]. Machine Learning (ML), on the other hand, seeks to control systems with minimal human intervention. Integration of ML and ITS provides adequate solutions for optimising traffic signal difficulties.

RL is a framework of ML, that is interactive with the environment and establishes the best policy for sequential decision-making in the various fields of sciences through trial and error method [4]. An RL agent verifies the status of the environment and therefore, carries out an operation. Any action performed earns a reward or penalty and this reward is determined by the environmental impact of the action. The goal of the agent is to learn the best selection technique to maximize the discounted cumulative reward by means of repeated ambient interactions. Generally, for a large scale road network, as state-action pairs extend exponentially, the difficulty of using RL in traffic signal management increases in an exponential manner. Deep Learning has been highly appreciated and combined efficiently with RL approaches to yield DRL [5] in order to solve this dilemma. DRL has been an effective solution for sequential decision-making control problems in recent years, and has shown an incredible success in complex, dynamic and high dimensional environments.

In the present work, efficiency of the Policy Gradient algorithm is tested in two real time dynamic road networks with multiple intersections in which traffic signal is connected to each intersection. The following are suggested contributions from the method proposed:

- An adaptive traffic signal management system based on DRL is proposed to monitor multi-intersection traffic signals as simultaneous control of all the traffic signals of the network permits more efficient traffic handling.
- The traffic flow is a sequential spatio-temporal data stream. The DRL agent uses this data stream of the traffic environment to coordinate the traffic signals on the intersections. In contrast, this understanding is indiscriminate in the sense in which the agent controls a single traffic signal.
- To precisely describe the temporal information of the traffic environment, a stack of five frames is used to define state representation. It encourages the agent to obtain more environmental knowledge, leading to a better choice of actions.
- The agent is trained using Policy Gradient algorithm in several Deep Neural Network (DNN) models such as Fully Connected Neural Network (FCNN), Convolutional Neural Network (CNN), Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU) to estimate the best action selection policy to ensure effective traffic signal management.
- The productivity of Policy Gradient algorithm is already analyzed in a static network in [6]. However, dynamic real time networks help to portray actual scenarios with its associated complexities which is not possible to mention in static network. So, in this work, the entire simulation experiment is done in two different real time dynamic road networks, which is downloaded from OpenStreetMap (OSM).

This paper is divided into four more parts. In section 2, research gaps in former researches are being discussed. Section 3 presents the present work followed by section 4 which describes the simulation results and discussion. Finally, section 5 concludes the research work.

2 RELATED WORK

Recent progress in RL and DNN configurations has shown promising results to solve large dimensional dynamic control problems. Influenced by these accomplishments, two types of RL are built in [7]- the deep policy gradients and the value-based policy, which can better predict traffic flow in an intersection. Agents receive a glimpse of the current status of an integrated camera and generate control signals in any state. The value-base policy agent first calculates the value of all control signal, but the agent selects the signal by direct observation. It then chooses the maximum value for the optimum control action. These approaches have produced promising results and have shown they can find more stable policies compared with the anticipated work in the optimisation of traffic signals.

In order to learn the optimal policy for traffic signal management in congested scenarios, a policy gradient approach is suggested in [8], based on periodic circumstances and on time baseline. It is applied to a highly dense roundabout and evaluated on real data-set which shows that the policy reduces overall waiting time and emissions significantly while avoiding traffic congestion. It has been also shown that the suggested time base policy gradient algorithm is better than the classical baseline. In this case, only the roundabout network was considered, where it is not clear whether the algorithm will be functioning well in a real time dynamic road network also.

A number of algorithms for DRL is proposed in [9] in order to design signal control. A DNN is set up to learn the Q function from the inputs that have been sampled. On the basis of the DNN, it is necessary to model the control activities and the change of the system states in the appropriate signal timing policies. Although there have been a few approaches towards approximation of the maximum discounted future reward, this method shows that DNNs provide a more powerful and convenient tool in order to achieve the goal. After setting up a number of policies, the explore-exploit dilemma will be faced and for this, possibly better policies may be opted in terms of exploiting the current working policy.

In [6], a single DRL agent uses the policy gradient algorithm to handle a traffic signal of several intersections. The agent is trained, in particular, on spatio-temporal environmental data to act in a variety of deep-neural networks. Three different simulation metrics are analysed for the simulation experiment. Policy gradient method is executed in various deep neural network models, namely, in order to bring about improved control of traffic signals. FCNN and CNN are used for approximating the action selection policy. But the simulation experiment has been executed in a static network here.

A promising and scalable multi-agent approach to DRL using the Deep Q Network (DQN) algorithm is introduced in [10]. It examines traffic signal control policies with the help of new rewarding features and it is suggested to combine the popular Deep Q learning algorithm with a coordination algorithm in order to manage traffic signals. It is demonstrated that this approach reduces travel times in relation to previous research on reinforcement methods. It works for multi-agent signal control earlier, but the DQN algorithm can oscillate, a problem which was also found in previous works on profound reinforcement learning.

3 PRESENT WORK

3.1 RL:

In recent years, RL techniques for controlling the traffic signals are being explored by researchers in order to mitigate traffic congestion. The purpose of RL is to learn an optimal policy starting from the initial state. An agent in RL interacts with the environment by trial and error method to learn the best policy for selecting an action for maximising the discounted cumulative reward in the longer term. The RL environment can be defined as a type of Markov Decision Process (MDP) which contains the tuple $< \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R}, \boldsymbol{\gamma} >$ where $\boldsymbol{S}$ is the state space, $\boldsymbol{A}$ is a set of actions, $\boldsymbol{P}$ is the state transition probability function, $\boldsymbol{R}$ is known as the reward function and $\gamma \in (0, 1)$ is a discount factor.

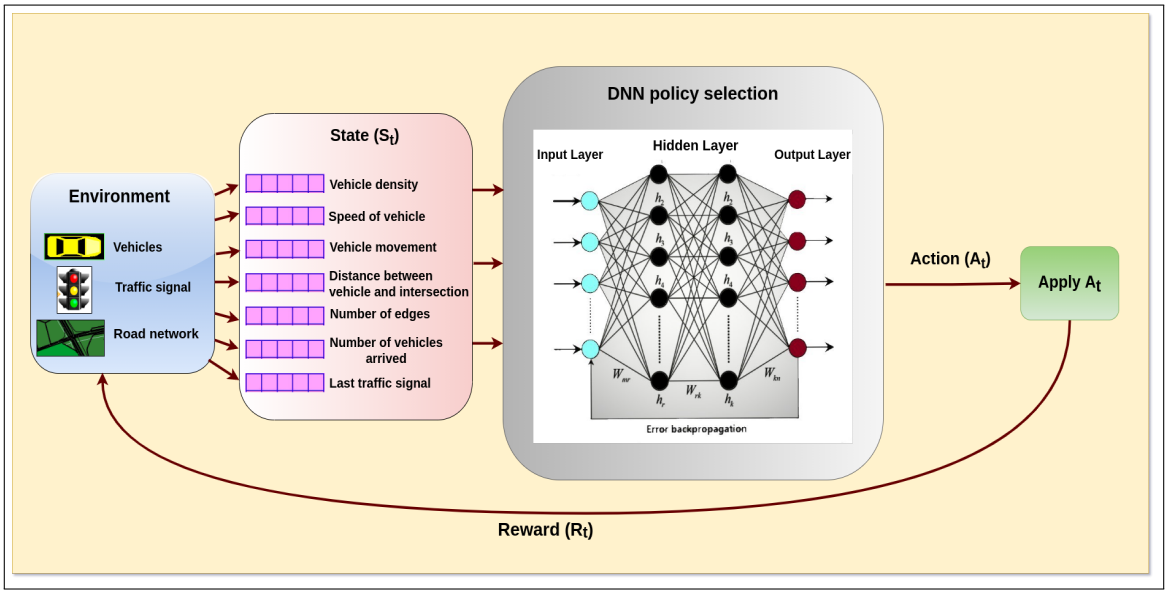

Fig. 1. Reinforcement Learning Framework

In the present work, two simulators Simulation of Urban MObility (SUMO) and Traffic Control Interface (TraCI) have been considered in order to create the environment and the state of the environment is not only dependent on traffic signals of the road network but also the vehicles present in it. At each discrete time steps t, the current state $s_t \in \boldsymbol{S}$ is received by the agent (Fig. 1). It then selects an action $a_t \in \boldsymbol{A}$ to perform and it is acted upon the environment. The agent obtains a reward R_t for its action and the environment gets moved to a new state $s_{t+1} \in \boldsymbol{S}$. If the agent's action leads to a favourable environmental reward, then the chance of carrying out such an action will be increased otherwise it will decrease. However, RL algorithms face challenges due to complex and enormous state-action pairs of vast road networks.

3.2 DRL:

In contrast to RL, DRL seems to be more powerful and stabilised, particularly for high-dimensional state-action problems. The behaviour of an agent is defined by a policy π ($\pi : \boldsymbol{S}$ X $\boldsymbol{A} \rightarrow \boldsymbol{R}$) in RL. In DRL, weights and biases of the neural network, which can also be called as a set of parameters θ, are described to parameterise the policy π, which is

defined as π_θ. A probability distribution over actions is returned by π_θ. The task of an agent is to follow a policy $\pi_\theta(a_t|s_t), a_t \in \boldsymbol{A}, s_t \in \boldsymbol{S}$. The agent is trained for numerous episodes (K), each of which has multiple iterations (T). The cumulative sum of discounted reward at t^{th} iteration of an episode (G_t) is defined as follows (Equ. 1):

$$G_t = \sum_{i=t}^{T} \boldsymbol{\gamma}^{i-t} R_i \qquad [11] \tag{1}$$

3.3 Policy Gradient:

Function approximations based on Neural Network are sufficient to make RL effective at high dimensional state spaces (i.e. a policy used to map input traffic to a signal traffic control measurement). It could either be accomplished using action-value methods or by directly learning a parameterised policy using policy gradient method by measuring the value of actions. The policy gradient algorithm is intended by a gradient-ascent approach to optimise the parameterised policy function $\pi_\theta(a_t|s_t)$ to maximise the expected value of G i.e. ($J(\theta)$).

As stated by the policy gradient theorem (Equ. 2), the derivative of the expected reward ($\nabla J(\theta)$) is the expectation of the product of the discounted reward (G_t) and gradient of the log of the policy $\pi_\theta(a_t|s_t)$.

$$\nabla J(\theta) = E_{(\pi,\theta)}[G_t \nabla log \pi_\theta(a_t|s_t)] \tag{2}$$

The parameters of the neural network are modified using the following update rule (Equ. 3) for each episode k, according to gradient ascent, until $J(\theta)$ is maximised.

$$\theta_{k+1} = \theta_k + \alpha \nabla J(\theta_k) \qquad [11] \tag{3}$$

3.4 The method framework:

The agent is trained using policy gradient algorithm which is outlined in Fig. 2. Let the number of intersections in a road network is N. Initially, the parameters (θ) of the neural network are set to random values using normal distribution and the replay memory (*EPISODE*), which serves as a buffer, is set to null. The current state of the environment at time step t is s_t which is sent to the DNN model. The DNN generates a probability distribution set (P_t) over actions for s_t. The P_t consists of the probability distribution of action at intersection n (p_t^n), $\forall n \in N$ (Equ. 4).

$$P_t = \{p_t^n, \forall n \in N\} \tag{4}$$

If the value of p_t^n is greater than 0.5 then the value of the action at n (a_t^n) is set as 1 otherwise 0. An action set (A_t) is formed which consists of $a_t^n, \forall n \in N$ (Equ. 5).

$$A_t = \{a_t^n, \forall n \in N\} \tag{5}$$

Subsequently, A_t is applied in the environment and reward R_t, next state s_{t+1} is received. Then, the tuple $< s_t, P_t, A_t, R_t, s_{t+1} >$ is stored in the buffer *EPISODE*. If one episode is completed i.e. all the T iterations of one episode are over, then for each tuple in *EPISODE*, the following steps are executed. Since, a single agent controls multiple intersections, the agent at first calculates the average binary cross entropy loss ($Loss_t$) $\forall N$ (Equ. 6).

$$Loss_t = \frac{-1}{N} \sum_{\forall n \in N} (p_t^n \; log_2 a_t^n + (1 - p_t^n) \; log_2 (1 - a_t^n)) \tag{6}$$