Reducing Congestion in an Intelligent Traffic System With Collaborative and Adaptive Signaling on the Edge

ABDUL JALEEL¹, MUHAMMAD AWAIS HASSAN², TAYYEB MAHMOOD³, MUHAMMAD USMAN GHANI², AND ATTA UR REHMAN³

¹Department of Computer Science, Rachna College of Engineering and Technology, Lahore 52250, Pakistan
²Department of Computer Science, University of Engineering and Technology Lahore, Lahore 54890, Pakistan
³Department of Electrical Engineering, Rachna College of Engineering and Technology, Lahore 52250, Pakistan

Corresponding author: Abdul Jaleel (abduljaleel@uet.edu.pk)

ABSTRACT The advancements in Edge computing have paved the way for deep learning in real-time systems. One of the beneficiaries is an adaptive traffic control system that responds to real-time traffic observations by governing the signal phase and timings. Reinforcement Learning (RL) is extensively utilized in the literature in order to decrease traffic congestion in a road network. However, most of the previous works leverage centralized and cloud-based RL due to the computational complexity of underlying deep neural networks (DNN). Therefore, a persistent challenge towards adopting Edge learning is in devising a Multi-Agent RL in which agents are simplified, and their state spaces are localized but they perform comparable to the centralized RL. This article presents a Collaborative and Adaptive Signaling on the Edge (CASE), a novel Multi-Agent RL approach to control the traffic signals’ phase and timing. Each signalized intersection in the road network is provided with an Edge Learning Platform which hosts an RL-Agent that observes local traffic states and learns an optimum signal policy. Moreover, CASE allows collaboration among RL-Agents by sharing their signal phase and timings to achieve convergence and performance. This collaboration is limited to one’s direct neighbours only to minimize the computational complexity. We performed rigorous evaluations in terms of the choice of RL methods and their state space/reward and found that our collaborative state-space has resulted in a performance comparable to a centralized RL yet with a cost similar to the decentralized RL. Finally, a performance comparison of the CASE controller ported to the state-of-the-art Edge learning platforms is presented in this article. The results show that the proposed CASE controller can achieve real-time performance when ported to a general-purpose GPU-based platform. This arrangement achieves more than 8 times improvement in computational time over conventional embedded platforms.

INDEX TERMS Traffic congestion, collaborative signaling, artificial intelligence, distributed computing, embedded machine learning, Proximal Policy Optimization, deep Q-networks.

I. INTRODUCTION
With the increasing number of vehicles on roads, traffic congestion is becoming a critical issue for big cities. The congestion increases travel time, fuel consumption and air pollution [1]. Due to congestion in large cities of North America, people travelled extra 6.8 billion hours annually and consumed 3.1 billion gallons of fuel, which raised the congestion cost to $153 billion [2] and emission of greenhouse gases caused global warming and health risks. Moreover, frequent waiting in traffic queues may lead to “Road rage”, an umbrella term used to describe a host of psychological disorders [3].

Countries spend billions of dollars on road expansions to avoid traffic congestion. According to the INRIX traffic scorecard report of 2017 [4], the UK has spent £500 million, Dallas has paid $1-billion, and Germany has invested 21 million Euros in road infrastructure. However, the development
of road infrastructure is not sufficient alone, in our opinion. Instead, recent developments in remote sensing and Information and Communication technologies [5]–[7] can be leveraged to address the traffic congestion problem.

The solutions provided by intelligent transportation systems to alleviate traffic congestion fall into two categories [8]. We can reroute the vehicles away from busy road segments and intersection [9] as adopted in computer networks. However, vehicles in most parts of the world lack equipment to receive such directions. Alternatively, adaptive traffic signaling can be employed based on the congestion statistics or real-time monitoring on the roads [5]. Optimizing the Signal Phase and Timing (SPaT) in an efficient, intelligent, and adaptive way can be more cost-effective to handle traffic congestion at signalized intersections [10].

Conventional SPaT schemes deliver a fixed timing of signal phases at an intersection based on historical traffic data records. Phase timing is embedded in the Onboard Equipment (OBE) of a traffic signal or in the Roadside Equipment (RSE) of the signalized intersection and can be periodically updated. An improvement over this simplistic system is a responsive control system that uses stored configurations of SPaT to adapt the signal timings based on real-time traffic environment. Split, Cycle and Offset Optimization Technique (SCOOT) [11] is deployed in Britain as a traffic control system that records traffic queues on the intersections. It continuously adjusts signal timings such that the sum of the queues is minimized in a specific area, based on a heuristic optimization algorithm that emulates adaptive SPaT plans by using TRANSYT, an evolutionary model using platoon-dispersion equations [12]. Alternatively, Sydney Coordinated Adaptive Traffic System (SCATS) developed in Australia does not require such models. Instead, it is a decentralized system that utilizes local controllers for traffic control [13]. A library of preset plans is provided to minimize vehicle stop times with light demand, minimize delay with normal demand, and maximize throughput with heavy demand. Local adjustments at the top of presets are allowed to adapt to the instantaneous traffic profiles. Other widely used responsive traffic control systems include RODYN and CRONOS developed in France, UTOPIA developed in Italy, and OPAC and RHODES developed in USA [14].

Because SPaT can be modeled as a Markov Decision Process (MDP), dynamic programming is applied to traffic control systems in many recent studies [6], [7], [15]–[17]. These dynamic programming-based schemes require a mathematical traffic model which is based on the vehicles’ parameters acquired from In-Vehicle sensors. However, traditional vehicles generally lack such sensors, advocating a model-free control technique such as Reinforcement Learning (RL). At the core of RL is an Agent that learns an optimum policy iteratively by choosing an action based on the observation of its state space such that a Reward is maximized. In SPaT control, the reward may be traffic throughput at an intersection, wait time, or another figure of merit. Therefore, many recent attempts are made to solve the question of optimal SPaT pattern with deep Q-learning and Policy Gradient methods [18]. The Deep Q-Network (DQN) is a commonly used deep Q-learning method. Whereas, one recent example of the Policy Gradient method is Proximal Policy Optimization (PPO) that solves the Partially defined Markov Decision Processes (MDP) where the state is not fully defined and/or observable and where the state observations are noisy, such as the traffic signaling problem [19].

Despite the advantages mentioned above, the application of RL methods is often limited in real-life systems due to the computational complexities of underlying Deep Neural Networks (DNN) [20]. The compute kernels of DNNs are computationally intensive; therefore, centralized and cloud computing resources equipped with deep learning accelerators (DLA) are often leveraged [19], [21]. In traffic control, however, a centralized cloud-based solution is deemed sub-optimal due to communication bandwidth and real-time latency requirements [20]. Alternatively, Fog and Edge computing have been proposed to alleviate these bottlenecks, especially in the context of real-time applications [22], [23].

This research focuses on the use of Embedded Deep Learning at the Edge of traffic control networks to alleviate the above-mentioned problems related to bandwidth and latency [24]. Because, the embedded platforms (with DLAs) are often resource-constrained to smaller DNNs [25], we argue that the RL must be simplified and truncated, for example, to the Signal control at a single road intersection, resulting in a decentralized Multi-Agent RL. However, previous studies in this domain show that the decentralized Multi-Agent RL may not converge well, especially in the traffic control problem, because an optimal signal control at one intersection depends on the signal state at other intersections [18]. In order to converge, these individual RL Agents must not work in silos. Instead, they should collaborate to maximize a joint reward by sharing their decisions [14]. Thus, there is a need to propose a decentralized and collaborative, multi-agent, Reinforcement Learning-based solution which can leverage contemporary edge learning platform to infer optimal signal phase and timing in a real-time traffic control system. To address this problem, we propose a Collaborative and Adaptive Signaling on the Edge (CASE) which is an efficient and scalable Multi-Agent RL-based traffic control system. To the best of our knowledge, CASE is the first attempt to deploy deep learning on Edge of a responsive and adaptive traffic control network.

We addressed the following research challenges to employ a Multi-Agent RL in real-time traffic control systems.

- Defining of an optimal state-space, e.g., consisting of vehicle queue length, traffic throughput, etc., that impact the RL performance.
- Choice of a suitable Reward function, e.g., average throughput or average waiting time experienced by the vehicles.
- Scale of Horizon: Ideally, a truly optimal SPaT pattern should consider all the intersections of a road network.
Alternatively, considering only local state-space at a road intersection is faster to process but may not converge well [26]. A balanced horizon is required that includes a subset of nodes that effectively reflect the system state.

Our CASE system is backed by a rigorous evaluation based on FLOW, an RL framework that uses the SUMO traffic simulator as an environment in which each signalized intersection is modeled as an RL Agent. We compared the results for a Fixed Time, a conventional adaptive system (SCOOT), and our DQN-based D-CASE and PPO-based P-CASE setups. In performance evaluations, the CASE system was deployed on an Edge Learning Platform as a part of Roadside equipment to infer the SPaT pattern for concerning intersection by using a collaborative state-space, including local traffic observations (from traffic cameras) and SPaT information from neighboring intersections. When CASE is ported to general-purpose GPU-based platforms, it achieved more than 8 times improvement in computational time for Nvidia’s Jetson Nano Development Kit over Raspberry Pi4 based conventional embedded platform. Reducing the horizon only to the neighbouring intersections has allowed real-time communication to be affordable for SPaT sharing in a Multi-agent RL. Contrary to previously proposed and adopted solutions, the CASE system focused on the average wait time of vehicles as RL reward, and thus reducing the congestion in a psychological sense and mitigating the “Road rage”.

The rest of this article is organized as follows. The next section presents the background, recent advancements, and motivation behind this work. Section III presents the CASE as our proposed edge-based deep RL system, highlights the methodology of our work, and introduces the RL algorithms we used for decentralized and collaborative traffic control. The experimentation and the results are presented in Section IV. Section V presents deductions based upon discussions on the results. Finally, Section V1 concludes the article and highlights our future work.

II. BACKGROUND AND RELATED WORK

At a traffic intersection, multiple traffic lights work in synchronization to manage the flow of traffic, where each traffic light switches among the signals of Red (R), Yellow (Y) and Green (G). The time duration of staying at a particular signal out of Red, Yellow, or Green is termed as one phase of the traffic light [10]. Conventionally, the green-light phase and red-light phase are managed for every traffic inflow arm, whereas the yellow-light phase is set as a transition time to the red-light phase are managed for every traffic inflow arm, thus reducing the congestion in a psychological sense and mitigating the “Road rage”.

The rest of this article is organized as follows. The next section presents the background, recent advancements, and motivation behind this work. Section III presents the CASE as our proposed edge-based deep RL system, highlights the methodology of our work, and introduces the RL algorithms we used for decentralized and collaborative traffic control. The experimentation and the results are presented in Section IV. Section V presents deductions based upon discussions on the results. Finally, Section V1 concludes the article and highlights our future work.

A. TRADITIONAL APPROACHES

A SPaT optimization depends on the data from sensors to observe the vehicles passing through the road intersections. Loop sensor-based detectors are traditionally used for vehicle detection; however, such sensors can only detect passing by vehicles. More recently, the use of cameras is becoming a new source of data to get more sophisticated vehicles related to data [27]. With advances in the image processing techniques and availability of high computation speed, Video data from the camera can be processed in real-time to extract a detailed representation of the traffic situation on roads [28]. Traditional SPaT techniques usually suppose that every intersection is independent of other intersections in a region and try to optimize the signal timings of every intersection independently. Such techniques develop a traffic model and
use them to calculate the cycle length of signals of an intersection. Webster [29] is one such technique that assumes that the traffic flow at an intersection is uniform for a certain duration. Based on this assumption, this technique calculates the intersection’s cycle duration and decides the phase split to minimize the travel time of all the vehicles at an intersection. GreenWave [30] is another traditional signaling technique that tries to reduce the number of stops for vehicles traveling in a certain direction. This is achieved by implementing the same cycle length at all the intersection. This method helps to reduce the stopping time of vehicles moving in a certain direction by providing them a green wave. This minimizes their number of stops and optimizes the unidirectional traffic, using offsets in SPaT. Another traditional approach, Maxband [31] provides a mechanism to reduce the number of stops for vehicles traveling in two opposite directions. This technique also implements the same cycle length on all the intersections.

B. ACTUATED AND ADAPTIVE APPROACHES

Actuated control decides a signaling plan based on the requests for a green signal from the current and other completing phases. Based on the distance of oncoming vehicles from a signal and the number of waiting vehicles, this technique decides whether the duration of green signals should be extended or not. A similar approach, Self-Organizing Traffic Light Control (SOTL) [32] decides the extension in the current green phase on the basis of the number of vehicles approaching an intersection. Max pressure control [33] balances the queue length between neighboring intersections by minimizing the pressure of the phases of an intersection, where the pressure is defined as the difference between the overall queue length on incoming approaches and outgoing approaches. Sydney Coordinated Adaptive Traffic System (SCATS) [13] works on the principle of taking predefined signal plans as input and iteratively selects from these plans.

SCOOT records traffic queues on the intersections and continuously adjusts signal timings such that the sum of the SCOOT records traffic queues on the intersections and continuously adjusts signal timings such that the sum of the traffic queues is minimized in a specific area [11]. Shen et al. [6] presented a dynamic speed-truncated normal distribution model and dynamic Robertson model with dynamics that outperforms the existing methods. In recent developments, vehicle infrastructure integration (VII) technology collects state space information about upcoming vehicles in terms of their location and speed, which is then used to manage the timing of a traffic signal [15]. The system works for individual intersections with less state-space parameters that can be enhanced and shared with nearby intersections to manage the traffic flow in a collaborative manner. Similarly, Yao et al. [17] used the technology of connected automated vehicles to obtain vehicles’ identification, position, speed, and acceleration in addition to traditional traffic data, and applied dynamic programming and most predictive model approach to reduce the traffic congestion. However, dynamic programming requires a model that is not readily available in traditional traffic scenarios and depends on parameters that explicitly require Vehicle to Intersection Communication, which legacy vehicles generally lack.

C. INTELLIGENT APPROACHES

Model-Free Reinforcement Learning is an answer to optimization problems of those Markov Decision Processes for which state space is not fully observable, or a mathematical model is not well-developed or well-understood. A good survey on the application of RL methods to the adaptive traffic control systems is presented in [18], [26]. Recently, Li et al. [34] claimed that DNNs could be used to learn the dynamics of a traffic system and define a signal plan by modeling the control action and system state. It could be efficient if the problem is mapped to reinforcement learning. A Q-learning based technique was also used in [35] to optimize the traffic signals, but each signal optimized itself without looking at the other signals’ policy. A multi-agent approach was used in [36] to avoid congestion, but the state parameters were not enough to attain a global optimum. The communication between the agents was another overhead. Similarly, two decentralized actor-critic algorithms were suggested by Zhang et al. [37]. In the actor step, an agent takes action without affecting the policy of other agents. In the critic step, the agent shears its value function to its near agents, which is used in the successive actor step. But, the recursive process has computation overhead. Li et al. [38] claimed mini-max multi-agent deep deterministic policy gradient for reinforcement learning performs best in cooperative and competitive scenarios. Liang et al. [39] proposed a deep reinforcement learning model to control the traffic cycle by getting the position and speed of vehicles from different sensors. However, the environmental observation was not enough to decrease the overall congestion, and agents make their policy without looking at the neighboring signal’s state. Natafgi et al. [40] implemented an adaptive traffic light system for one isolated intersection considering queuing times and queue length. However, the real environment consists of multiple intersections. So, to resolve the overall traffic congestion problem in a region, there must be a collaborative state-space that carries an environment observation with neighboring signal’s state information.

D. ORGANIZATION AND ARCHITECTURES

In order to effectively respond to real-time traffic, close interactions between intersections are pivotal. In addition, all the interactions over the network must be synchronized. On the other hand, prior works in decentralized traffic control suffer due to partial observability of the whole transportation network as well as its high dynamic traffic patterns. Robertson and Bretherton [41] used a distributed control method but assumed that the sensor information within the whole area was easily obtained from centralized servers. This may suffer a communication bottleneck in a huge urban area. Shenoda [42] assumed that the coming vehicles follow a Poisson distribution to build a decentralized coordination algorithm. Xie et al. [43] used fixed traffic signal phases to solve the
conflicting traffic flows, which is not flexible to optimize concurrent traffic flows within the intersections.

A centralized deep learning model forces multiple participants to pool their data in a centralized server to train a global model on the combined data [10]. However, this centralized Cloud-based machine learning required enormous bandwidth and computing resources. To accelerate inference, [44] proposed a distributed DNN (DDNN) architecture across the Cloud, the Fog, and the Edge devices and allowed fast inference on end-devices and complex inference on the Cloud. Li et al. [24] combined Fog computing with deep learning by dividing the pre-trained Cloud-level model into two parts: the lower layers near the input data are deployed into Fog nodes, and higher layers are kept into the Cloud. However, they are keen on deploying a pre-trained model to offload processing during inference while neglecting the computation-intensive training process.

From the discussion in this section, which is also summarized in Table 1, it can be deduced that RL is the most suitable choice when optimizing the Signal Phase and Timing (SPaT) on the signalized intersections with traditional vehicles. RL does not require a traffic model (is Model-Free) and is able to learn a better SPaT policy based on partial state observations. Moreover, Multi-agent RL with decentralized computing resources is well suited in road networks, and individual agents may learn control policy based on the decision of their neighbouring agents. Finally, the advent of Edge-based Deep learning platforms enables and motivates us to propose our Collaborative and Adaptive signaling on the Edge (CASE), which is detailed in the next section.

**III. PROPOSED SOLUTION**

This research proposes a low-cost hardware/software architecture to minimize traffic congestion through an intelligent traffic system with a Collaborative and Adaptive Signaling on the Edge (CASE). In Fig. 2(a), a signalized intersection is depicted, highlighting a Roadside Equipment (RSE) in Gray, which is connected to four Onboard Equipments (OBE) in Brown, through Power line communication channels. The enlarged view of one intersection is presenting the detailed architecture of CASE, which consists of the following subsystems.

**A. LANE OBSERVATION AND SIGNAL CONTROL (LOSCON) SUBSYSTEM**

LOSCON constitutes the OBE which is usually mounted on the Signal Pole and has three main components, the Signal Lights Controller (SLC), the Lane Observation Camera Interface (LOCI), and Communication Interface (CI), as shown in Fig. 2(d). SLC implements the necessary power electronics circuits to drive the traffic lights. LOCI interfaces with a traffic camera that captures the live traffic video on the lane and feeds it to LOCI’s pre-processing vision algorithm that extracts road statistics such as traffic density and queue lengths at the intersection [27], [28]. CI enables transmission of the above records to the CASE controller and receives commands related to SLC. Powerline communication (PLC) is leveraged in this work for data communication because the same cables can be used for delivering power as well as communication packets.

**B. CASE CONTROLLER**

The CASE controller is housed in the Roadside Equipment (RSE), as shown in Fig. 2(c). It consists of a SPaT Controller, an Embedded Learning Platform (ELP), and a Communication Interface (CI). SPaT Controller implements the baseline Signal Phase and Timing pattern. It stores a preset phase sequence of an intersection, with minimum and maximum phase timing constraints at that intersection. Collaborative and adaptive, RL-based algorithms are implemented in ELP. It consists of an embedded platform with the required computing capabilities to run supported deep learning algorithms. In this work, we utilize and compare three platforms, a Raspberry-Pi 3 based Single Board Computer, a Raspberry-Pi 4 based Single Board Computer and a Nvidia Jetson Nano development kit. The compute capabilities of these boards are presented in Table 3. The boards are low-cost edge platforms, specially designed to cater to high performance and low power embedded control and learning applications. Based on real-time traffic data communicated form local LOSCONs and signal states from neighbouring CASE controllers, an ELP finds an optimum split (phase duration) of the signal phases each SPaT cycle, through its RL agent. The split and phase duration are passed to the SPaT controller that orchestrates the LOSCONs’ phase and timings.

The overall process flow of SPaT control is given in Fig. 3. In Step 1, a traffic camera feeds live traffic video to LOSCON, which extracts the estimated state observations. The recorded traffic states from $N$ LOSCONs and $N$ neighbouring intersections are sent to the CASE controller in Step 2. In Step 3, the RL Agent in ELP infers SPaT for the next phase cycle, based on its learned policy. SPaT is then sent to the SPaT
controller (in Step 3a), which transmits signals control commands and timing to the connected LOSCONs. ELP also sends SPaT to neighbouring intersections (in Step 3b) to influence the actions (SPaT inference) at their ends. Each SPaT controller then commands the attached LOSCONs with their respective Phase timings (Step 4). Finally, in Step 5, an SLC (in LOSCONs) controls the attached traffic lights, as commanded by the SPaT controller in the previous step.

**C. RL Agent**

The RL Agent is the brain of the CASE controller and continuously processes the collected statistics through Reinforcement Learning (RL). It models the SPaT problem as a Markov Decision Process (MDP), which requires a state space, an action space, state transition probability function, and an immediate reward function resulting from a transition. In this work, we utilize a simulator, SUMO, as an RL environment. SUMO is responsible for traffic simulation and feeds the RL Agent with relevant state observations, which also help to compute step rewards resulting from those actions which the RL agent confers in a step. Actions are then communicated to SUMO. Here, a step is a SPaT cycle at the concerned intersection. Therefore, in each RL step, the RL agent sends a SPaT message to SUMO, which includes explicit timing of all phases, predefined in the SPaT controller, and SUMO returns observations of the consequent traffic state, as shown in Fig. 4.

1) **Collaborative State-Space**

The collaborative state space of the RL agent includes local information, including average queue length and vehicle density observed at an intersection when a signal is red, and...
Algorithm 1: Deep Q-Network Based Adaptive Policy Estimation

| State-Space | Input: $s_t$, Action Set $a_t$, Time $t$ | Output: Reward $r_t$, Optimal Policy $\pi : s \rightarrow A$, V-Values $Q_t(s_t, a_t, \theta)$. |
|-------------|----------------------------------|--------------------------------------------------|
|             | procedure OptimalPolicyDQN($s_t$, $a_t$, $t$) | $Q_t(s_t, a_t, \theta) = R_t + r \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta)$ |
|             | 3: Store transition in replay buffer as $<s, a, r, s'>$ | |
|             | 4: Calculate the Temporal Difference | $TD_t(a_{t+1}, s_{t+1}) = R_t + r \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta) - Q(s_t, a_t, \theta)$ |
|             | 5: After every k steps, copy Q-Value DNN weights to the Action DNN weights. | |
|             | 6: Observe the state for the best-predicted reward against each action. | |
|             | end procedure | |

Therefore, RL action is a vector $A = (t_A, t_B, t_C, t_D, t_E, t_F)$. In CASE, this action is encapsulated in a SPaT message, which is communicated to the local SpaT controller and the neighbouring CASE controllers.

3) REWARD FOR THE AGENTS’ ACTIONS

In this work, we consider the average waiting time of vehicles in a traffic queue to compute RL Reward. A vehicle experiences waiting time when it arrives and enqueues at a road link. The waiting time is zero when it arrives and leaves the road link during the Green signal. Alternatively, it may wait for a fraction of the SPaT cycle or multiple cycles at highly congested links before crossing the intersection. Average wait time at one road link is the average of wait times experienced by all vehicles using that road link and is an instantaneous value that is difficult to record. However, using the average wait time for RL Reward has its merits, which is evaluated later in this article. A method to estimate the same is also presented. CASE strives to maximize the reciprocal of this wait time as a reward. For quantization in discrete algorithms, it also multiplies this number by 255, to form a quantized (8-bit) reward.

4) DEEP Q-NETWORK BASED REINFORCEMENT LEARNING

The first approach towards an RL Agent is to employ Deep Q-Network (DQN), an RL algorithm that was used by many researchers [18] to solve traffic congestion problems. The DQN applies a DNN to approximate the action-value (Q) function, where the action chain determines the policy $\pi$. It uses the replay buffer to store past experiences for later experience replay. The state observations become the input of the DNN. The output is the approximate Q-value of each action that an agent can perform. The DNN finds Q-values by using (1), as adopted from [45].

$Q_t(s_t, a_t, \theta) = R_t + r \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta)$  \hspace{1cm} (1)
Algorithm 2 Proximal Policy Optimization Based Adaptive Policy Estimation

**Input:** initial policy parameters \( \theta_0 \), initial value function parameters \( \phi_0 \)

**Output:** Determines the expected reward for a future policy.

1: **procedure** ProximalPolicyOptimizer(Spacket, Time)  
   2:    ▷ Determine Optimal policy
   3:     for \( n = 0 \) to \( n \) do
   4:         Collect the set of trajectories \( D_k = \{\tau_i\} \) by running policy \( \pi_k = \pi(\theta_k) \) in the environment.
   5:         Calculate reward \( R_k \)
   6:         Advantage estimation, \( A_t \), based on the current value function.  
   7:            ▷ using any method of advantage estimation
   8:         The policy is updated on maximizing the PPO objective, via stochastic gradient ascent with Adam, over a learning rate of 0.001
   9:         \[ \theta_{k+1} = \arg \max_{\theta} E_{s,a} Q(s, a, \theta_k) \]
  10:     end for
  11: **end procedure**

Here, \( R_k \) is the reward that an RL Agent gets on choosing the best action which gives the maximum \( Q \) value of subsequent state as represented by \( \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta) \) where, \( s \) is the set of state-space parameters for the environment observation at the time step (\( t = 0, 1, 2, 3 \ldots \)), whereas, \( A \) is the possible set of actions that the agent can take. For a state \( s_t \), an agent takes the action \( a_t \in A \) and receives the reward \( r_t \). The agent’s objective is to maximize the reward while finding an optimal policy \( \pi : s \rightarrow A \).

We used the temporal difference to update the agent’s knowledge at each time step \( t \). The temporal difference makes the agent learn about every action that is taken and governed by (2) as given in \([45]\).

\[ TD_t (a_{t+1}, s_{t+1}) = R_t + r \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta) \]

By substituting (1) into (2), we get the learning of the Q-function \([45]\), as given in (3).

\[ Q_t (s_t, a_t, \theta) = Q_{t-1} (s_{t+1}, a_{t+1}) + \alpha TD_t (a_t, s_t) \]  

Training of DQN minimizes the error of temporal difference, which enables the DNN to select the best action selecting policy. The DQN used in this work is listed in Algorithm 1, whereas its architecture is illustrated in Fig. 4.

5) **PROXIMAL POLICY OPTIMIZATION (PPO)**

PPO maintains two separate policy networks, the current policy, as given in (4), and the policy established previously from experiences, as given in (5).

\[ current\_policy = \pi_\theta (\frac{a_t}{s_t}) \quad (4) \]
\[ previously\_established\_policy = \pi_{\theta_0} (\frac{a_t}{s_t}) \quad (5) \]

Here, \( a_t \) represents the action, i.e., the SPaT at state observation \( s_t \). The \( s_t \) consists of state observations, whereas \( \theta \) is the policy parameter, and \( \theta_k \) is the old policy. The next set of actions (i.e., policy) can be determined by (6), which we designed by following the work of \([47]\).

\[ \theta_{k+1} = \arg \max_{\theta} E_{s,a} Q(s, a, \theta_k, \theta) \quad (6) \]

Here, \( L \) denotes the empirical expectation over the time step, \( s \) consists of the state-space parameters, \( a \) is the Action, \( \theta \) is the current policy and \( \theta_k \) is the old policy. We used the PPO algorithm that takes the ratio of a new policy with the learned policy to ensure the steps that are close to the old policy. The formulation is given in (7), prepared in accordance to \([47]\).

\[ r_t (\theta) = \frac{\pi_\theta (\frac{a_t}{s_t})}{\pi_{\theta_0} (\frac{a_t}{s_t})} \quad (7) \]
The loss function used for reward calculation in the PPO algorithms is given in (8), designed accordingly of [47].

\[
L(s, a, \theta_k, \theta) = \min \left( \frac{\pi_\theta(a|s)}{\pi_\theta_k(a|s)} A^{\pi_\theta_k}, \right. \\
\left. \text{clip} \left( \frac{\pi_\theta(a|s)}{\pi_\theta_k(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_\theta_k(s,a)} \right) 
\]

(8)

Here, the second term clips the probability ratio if it moves outside the interval \(1 + \epsilon\) and \(1 - \epsilon\). The check function is given in (9), and we set the value of \(\epsilon\) at 0.3 in this work.

\[
g(\epsilon, A) = \begin{cases} 
(1 + \epsilon) & A \geq 0 \\
(1 - \epsilon) & \text{otherwise}
\end{cases}
\]

(9)

The advantage \(A\) is calculated in (10) as the difference of the weighted sum of all the rewards which the agent gets during each time step of the current episode.

\[
A = \text{DiscountedSumReward} - \text{BaselineEstimate} \quad (10)
\]

Here, DiscountedSumReward represents the result, and BaselineEstimate is denoting the value function.

The advantage function is depicted in Fig. 5. A positive reward means that the action is better than expected, and whenever the agent is in this state, it will increase the action’s probability. A negative reward means that action is worse than expected, and whenever the agent is in this state, it will decrease the action’s probability. The pseudocode of PPO, as used in this work, is listed in Algorithm 2.

IV. EXPERIMENTATION AND RESULTS

We evaluate the proposed CASE controller extensively through the experimental framework consisting of Simulation of Urban Mobility (SUMO), which is an open-source and portable project and allows microscopic simulation of multimodal traffic, vehicle communication, autonomous vehicles, and traffic management [48]. SUMO enables us to build realistic demand profiles on the top of actual road networks, which can be imported in major open and proprietary formats. Moreover, SUMO supports a Traffic Control Interface (TraCI) that gives access to running traffic simulation in a client-server fashion. TraCI allows state-values (observation) retrieval related to lanes, vehicles, sensors, etc., and provides control commands (targeting Traffic Lights, Connected Vehicles, etc.) through a C++/Python Application Programming Interface (API) that supports both Request-Response and Subscribe-Publish methods. The former method enables command/retrieval at one time, initiated with the client’s request, whereas the latter allows periodic information retrieval.

A. EXPERIMENTAL SETUP

RL solves Markov Decision Process with an agent and an environment. We employ SUMO as our traffic environment, whereas the RL Agents are modeled in FLOW [46], a framework to glue the traffic simulation environments with Ray RLLib, a library of RL algorithms [49]. The environment itself was built on the top of OpenAI Gym [50], an open-source RL framework. The whole setup is depicted in Fig. 6(a). The road network, traffic generator, and Signalized intersections are modeled in SUMO, and TraCI is exploited for the environment interface of Gym. The RL Agents employ DQN and PPO algorithms implemented in Ray RLLib while FLOW initializes, configures, and connects these components.

1) ROAD NETWORK AND TRAFFIC GENERATION

We created a mesh of \((3 \times 3)\) nine road intersections in SUMO. A sub-part of the arrangement is shown in Fig. 6(b), where each intersection is at the junction of two crossroads. The traffic lights on each intersection cycle through six

![Reinforcement Learning Simulator (OpenAI Gym)](image)

**FIGURE 6.** (a) Experimental Setup based on FLOW [46], and (b) SUMO generated road map showing six signalized intersections where Traffic Lights are emphasized for clarity, and traffic phases are annotated with arrows for understanding.
TABLE 2. Three State-spaces used in experimentation.

| Legend          | Type            | Observation method              | Parameters                                                                 |
|-----------------|-----------------|---------------------------------|-----------------------------------------------------------------------------|
| S1              | Intelligent,    | Video Camera, Neighbouring SPA,T| Vehicle queue length, Vehicle density, Vehicle throughput, Vehicle average velocity |
|                 | Collaborative   | message                         | Neighbouring signal phase and timing                                       |
| S2              | Intelligent,    | Video Camera                    | Vehicle queue length, Vehicle density, Vehicle throughput, Vehicle average velocity |
|                 | Isolated        |                                 |                                                                             |
| S3              | Conventional,   | Loop sensors                    | Vehicle throughput, Vehicle average velocity                               |
|                 | Isolated        |                                 |                                                                             |

FIGURE 7. D-CASE training performance in terms of Episodic Reward of State-spaces S1, S2, and S3 (Table 2). (a) Episode Average Reward, (b) Episode Maximum Reward.

phases, as shown in Fig. 1. The minimum and maximum phase timings are constrained at 10 seconds and 90 seconds, respectively. A customized traffic demand is generated for each experiment by using SUMO’s built-in demand generator algorithms through the FLOW interface.

2) RL ENVIRONMENT AND AGENTS

Logically, the LOSCONs at an intersection enforce the SPA,T at the start of a cycle, and the traffic state is sampled during the progress of that cycle. At the end of the cycle, the sampled state observations are communicated to the CASE controller, which in turn sends a new SPA,T packet to LOSCONs for enforcement, as well as to neighbouring CASE controllers for collaborative processing. With the local state observation as well as neighbouring SPA,T actions, a CASE Controller finds its own SPA,T action for the next cycle based on its learned policy, through Reinforcement Learning. Practically in this work, we simulate the aforementioned flow as a Multi-Agent Reinforcement Learning (MARL) in our experimental setup. Because the phase sequence is fixed and phase timings are constrained, no traffic rules are violated even when SPA,T cycles at intersections are asynchronous and RL Agents are in their learning phase. Three different state-spaces (S1, S2, and S3) given in Table 2 are employed, whereas DQN and PPO determine the RL policy in D-CASE (DQN-based CASE) and P-CASE (PPO-based CASE) setups, respectively. The average wait time at an intersection is chosen as a Reward and forms the basis of the accuracy of the learning algorithms. We divide a learning iteration into 20 episodes while each episode is 1000 Time steps long in D-CASE and 4000 Time steps long in P-CASE. The training phase consumes 240 iterations, and mean and maximum episode rewards are plotted as the learning curves.

3) REWARD APPROXIMATION

In RL, the mostly used reward functions are a weighted average of certain or all observable states of the environment. However, the learned behavior dramatically changes for slight modifications in weight functions. Using a simulator helps to learn the reward function itself. For example, in this work, the average wait time is used as a figure of merit and as a reward. However, wait time is not readily available in real-life traffic systems. In order to use this feature as a reward function, we utilize SUMO to generate local state observations (the State-space S2) and use a DNN as a function approximator that is trained with this generated data (Supervised training). The trained DNN for the reward is then used in the CASE controller. The average wait time on an intersection in seconds is divided by 255 (to aid quantization), so that a maximum reward represents a minimum wait time.

B. PERFORMANCE OF D-CASE

Fig. 7 depicts the average and maximum episodic rewards of each iteration for the three-state spaces, S1, S2, and S3. As the DNN of Q-value and Action-value learn with each iteration, the reward increases and saturates after 60 iterations. Beyond that, no significant improvement in the episodic reward is observable. Moreover, a collaborative state space such as S1 is significantly rewarding as compared to non-collaborative and reduced state spaces with partial local observations because the CASE controller decreases the vehicle waiting time by considering neighbours’
signal states. Moreover, a minimum temporal difference error in DQN means the agent chooses the best action at a given state. Less temporal difference ensures the best action selection policy. Fig. 9(a) shows the temporal difference error across the range of iterations. D-CASE achieves minimum temporal difference error with S1 state space which enables the underlying Q-networks to adopt an optimum SPaT policy.

C. PERFORMANCE OF P-CASE
When applied PPO for RL Agent, the learning results are plotted in Fig. 8. Again, the collaborative state-space S1 maximizes the RL reward as compared to S2 and S3. To adopt a policy that maximizes the long-term rewards, the RL Agent in P-CASE relies on the state transition and action probability distributions. Entropy regularization was used to improve policy optimization, which emphasizes exploration by encouraging the selection of more stochastic policies. We also measured the entropy of PPO to determine the unpredictability of actions in each policy. By visualizing the entropy curves in Fig. 9(b), we found that using collaborative (S1) state-space took less random actions, leading to the maximum future reward. If we compare the learning style of D-CASE with P-CASE, it is evident from the results that P-CASE gains higher reward (both average and maximum) after 100 iterations, whereas D-CASE achieves lower overall rewards in our simulations.

Finally, a custom demand pattern was generated at one road link of an intersection as a function of time, as in Fig. 10. The resulting phase split as Green time at the road link is also plotted. The phase split increases with the traffic density and achieves the goal of an adaptive and responsive traffic system. Minor variations in average waiting time with respect to demand are mainly due to factors including variations in state-space observations at other road links.

D. TRAFFIC THROUGHPUT AND VEHICLE WAIT TIME OPTIMIZATION
In this work, we qualify a traffic flow and traffic congestion with the throughput at a signalized intersection when signals go Green and the time a vehicle experiences waiting for the Green signal when the signals go Red. As a baseline,
A. Jaleel et al.: Reducing Congestion in an Intelligent Traffic System With CASE

FIGURE 10. Phase split (as the average Green time) at one road link of a signalized intersection. The traffic demand is shown as the number of vehicles incoming on the road link, while the average demand on other links of the intersection is held fixed at 30 vehicles per link.

we select a uniform and fixed timing (FT), of 30 seconds Green time, at each intersection. As a representative of a conventional adaptive systems [11], [13], [32], [33], we adopt a version of SCOOT that adjusts the Green signal time at a lane/road with fixed increments (of 5 seconds), based on its queue length. For comparison, we select the proposed CASE controllers with D-CASE and P-CASE agents. Collaborative state space (S1) is used in these experiments. The outcomes are plotted in Fig. 11 as throughput and average wait times, respectively, against increasing average queue lengths (normalized with respect to the lengths of the respective road links).

The trends of traffic throughput indicate a saturation of the number of vehicles passing through an intersection. As the average queue length on all road intersections is proportional to the instantaneous number of vehicles on the road network, throughput increases under light traffic conditions because more vehicles pass through an intersection. However, the throughput saturates due to the capacity of the road network and even decreases under heavy congestion because more vehicles are now idle, waiting for the Green phase. This is clearly visible in FT throughput and also in SCOOT that slowly increases the phase timing with respect to the queue length. RL Agents are better at dealing with heavy traffic because they strive to minimize the average waiting time experienced by the vehicles on the road network. However, maximum throughput is a function of road capacity and cannot be increased without improvements in the road infrastructure (e.g., the number of lanes or signal-free corridors) and current traffic laws (e.g., maximum vehicle speed).

Targeting the average waiting time of vehicles as an RL reward has advantages, however. Because waiting time experienced by a vehicle is an accumulated factor, optimizing this factor will automatically improve the throughput. However, the converse is not true, as by optimizing throughput, some road links with sparse traffic may experience unfairly long Red phases. Nevertheless, the increased congestion does increase the waiting times because vehicles may have to stay a couple of SpAT cycles on very busy links. This is evident in Fig. 11(b), where fixed timed signals yield dramatically increased waiting times under traffic congestion. SCOOT helps to relieve the pressure on such road links, but the proposed CASE controller clearly outperforms these traditional methods by significantly reducing observed wait times. In particular, P-CASE achieved a minimum average waiting time due to its superior environment learning while maintaining an appreciable throughput profile.

E. COMPARISON OF CENTRALIZED, DECENTRALIZED AND COLLABORATIVE SIGNALLING METHODS

To ascertain the effectiveness of collaborative Multi-Agent RL in CASE, we simulated a centralized, single-agent RL traffic scenario, as proposed in [36]–[38]. We used the same 3 × 3 (nine roads) intersections with similar traffic demand profile as in throughput and wait time experiments. But, we used a state observation vector consisting of an aggregate of individual state spaces S2, only. A SpAT action state of all the intersections is simultaneously emitted by this model. To reduce complexity, all intersections use aligned SpAT cycles of the same length (are synchronous). A decentralized and distributed (non-collaborated) multi-agent RL methodology, as proposed in [35], [39], [40] was also com-

FIGURE 11. (a) Throughput as an average number of total vehicles passing through a signalized intersection, and (b) Average waiting time in seconds at a signalized intersection, with respect to average queue length at one road link.
pared. Each decentralized intersection housed an individual RL Agent with S2 state space, only. The average waiting time is plotted in Fig. 12. In comparison with the decentralized RL, the CASE’s collaborative RL is effective in acquiring the quality of results, which is much comparable to a computationally-intensive centralized RL. Furthermore, CASE offers better scalability through its Edge Learning Platforms because a new intelligent signalized intersection does not require re-learning of a new, enlarged state-space, and is dependable because of no single point-of-failure.

F. PERFORMANCE OF EMBEDDED LEARNING PLATFORMS

We listed the details of three selected embedded systems for the choice of ELP in Table 3, highlighting their CPU, memory, GPU, and compute capabilities in terms of Giga Floating-point Operations per second. Raspberry Pi boards are popular in the Camera/Vision processing, whereas NVIDIA’s Jetson Nano board is equipped with CUDA cores, specifically provided to run compute-intensive DNN kernels. Both platforms support Linux distributions specifically tailored to the board capabilities and support the Keras/TensorFlow framework, which is employed in this work as an underlying software kernel of DNNs [25]. To compare, we executed a pre-trained version of P-CASE on these platforms and reported the average SPaT calculation time as the response time in Table 3. Due to CUDA Compute Capability, Jetson Nano board outperforms traditional embedded platforms in machine learning applications with DNNs. Therefore, this platform is recommended as ELP in the CASE controller.

V. DISCUSSION

In the domain of Reinforcement Learning, DQN and PPO are representatives of action-value and policy gradient methods. In Section IV, we compared both methods in the context of the SPaT problem. We found that PPO, along with a collaborative state space (S1, given in Table 2), minimizes the wait time experienced by vehicles by maximizing RL reward as evident in Fig. 8. We then compared these RL methods with a Fixed-timed control (FT) and a representative of the conventional adaptive techniques (SCOOT). Based on the results in Fig. 11, we found that RL-based techniques work best in scenarios with high traffic congestion. In particular, our proposed method (P-CASE) reduces the average wait time by half as compared to the Fixed-Timed SPaT, when the congestion (marked by average queue length) is as high as 80% (Fig. 11b).

The body of the previous research efforts in intelligent traffic control is dedicated to a centralized control where a holistic state space of a regional road network is acquired to train RL’s internal DNNs. We compared the P-CASE performance to a centralized Single Agent RL (with PPO and holistic Intelligent State Space S2) and a Decentralized and Isolated Multi-Agent RL (with PPO and holistic Intelligent State Space S2). We found that P-CASE achieves similar wait time reduction as compared to a centralized control with costs comparable to the decentralized control (Fig. 12). These costs include bandwidth requirements to collect traffic statistics in real-time and to disperse SPaT messages to signal controllers at intersections. In addition, the computational complexity of centralized methods is expected to scale up when more intersections are added to the road network. Finally, the addition of a new input would require rebuilding the underlying DNNs, followed by the computation-hungry process of re-learning. Therefore, it can be deduced that centralized control is not scalable in urban traffic control problems. On the other hand, a decentralized control where each intersection controls its own SPaT, is trivially scalable. However, a multi-agent RL with agents in silos is inefficient at best and unstable at worst in traffic control because traffic incident on an intersection depends on the signal state of the neighbouring intersections. Therefore, it is imperative to share the SPaT decisions among the neighbouring intersections so that individual RL-Agents may converge to a global reward. This deduction is also supported by the performance of P-CASE in Fig. 12.

Once we established the applicability of Multi-agent RL, we ported the proposed CASE controller to the contemporary Edge Learning Platforms to analyze and evaluate its real-time performance. The inference time of P-CASE (PPO with collaborative state-space S1) is tabulated in Table 3.
We find that Deep Learning Acceleration offered by the Jetson Nano Development kit is key to real-time performance, which is requisite in a traffic control problem. Moreover, the reduced state-space as compared to one required in a centralized control is bandwidth-efficient. Due to the reduced size of underlaying DNNs, Nano ELP was able to complete an inference (computing SPaT based on inputs as in Fig. 3) in 107ms on average. To compare Raspberry Pi3 and a newer and more resourceful Raspberry Pi4, ELPs rendered the inference time of 1560ms and 906ms, respectively. Therefore, based on our rigorous evaluations and discussions above, the proposed Collaborative and Adaptive Signalling on the Edge constitutes a real-time, intelligent, and a scalable traffic control system that greatly reduces the average wait time under congestion in urban road networks.

VI. CONCLUSION AND FUTURE WORK

We have proposed a Collaborative and Adaptive Signalling on the Edge (CASE) which employs an Edge computer to accelerate its compute kernels for solving the SPaT problem in a region. Based on results and discussions, we conclude that our proposed decentralized and collaborative Multi-agent RL (P-CASE) is more cost-effective and efficient to reduce traffic congestion on traditional signalized intersections as compared to the conventional fixed-timed and adaptive solutions. Although evaluations are performed offline with a traffic simulator in this work, we ported RL algorithms to ELPS and found that embedded deep learning accelerator solutions, like NVIDIA Jetson Nano, greatly reduce the computations time incurred in RL algorithms. A future work will evaluate the real-time performance of the proposed CASE framework at real-life signalized intersections, to further assist the deployment of Edge-based deep learning technologies into the adaptive control of traffic systems.

REFERENCES

[1] R. Fernando, “The impact of planned special events (PSEs) on urban traffic congestion,” ICST Trans. Scalable Inf. Syst., vol. 6, no. 23, Jul. 2018, Art. no. 159624.
[2] D. Schranks, B. Eisele, and T. Lomax, “2014 urban mobility report: Powered by Inrix traffic data,” Texas A&M Transp. Inst., Texas A&M Univ. Syst., College Station, TX, USA, Tech. Rep. SWUTC/15/161302-1, 2015.
[3] A. A. Ayar, “Road rage: Recognizing a psychological disorder,” J. Psychiatry Law, vol. 34, no. 2, pp. 123–150, Jun. 2006.
[4] G. Cookson and B. Phulse, “Inrix global traffic scorecard,” Inrix Res., Kirkland, WA, USA, Tech. Rep., 2017. [Online]. Available: https://www.veerkeerseunde.nl/Uploads/2017/2/INRIX-2016-Global-Traffic-Scorecard-Report-Appendices—English-FINAL.PDF
[5] Z. Cao, S. Jiang, J. Zhang, and H. Guo, “A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion,” IEEE Trans. Intell. Transport. Syst., vol. 18, no. 7, pp. 1958–1973, Jul. 2017.
[6] L. Shen, R. Liu, Z. Yao, W. Wu, and H. Yang, “Development of dynamic platoon dispersion models for predictive traffic signal control,” IEEE Trans. Intell. Transport. Syst., vol. 20, no. 2, pp. 431–440, Feb. 2019.
[7] L. Yang, Y. Wang, and Z. Yao, “A new vehicle arrival prediction model for adaptive signal control in a connected vehicle environment,” IEEE Access, vol. 8, pp. 112104–112112, 2020.
[8] A. L. Baztan, D. De Oliveira, F. Klügel, and K. Nagel, “To adapt or not to adapt—consequences of adapting driver and traffic light agents,” in Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning, Berlin, Germany: Springer, 2005, pp. 1–14.
[9] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, “Improving the efficiency of stochastic vehicle routing: A partial Lagrange multiplier method,” IEEE Trans. Veh. Technol., vol. 65, no. 6, pp. 3993–4005, Jun. 2016.
MUHAMMAD AWAIS HASSAN received the M.S. and Ph.D. degrees in computer science from the University of Engineering and Technology, Lahore, Pakistan, in 2009 and 2016, respectively. He is currently working as an Assistant Professor with the Computer Science Department, University of Engineering and Technology. His research interests include artificial intelligence, reinforcement learning, adaptive eLearning systems, and affective computing.

TAYYEB MAHMOOD received the B. degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 2006, and the Ph.D. degree in information and communication engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2013. He is currently working as an Assistant Professor with the Department of Electrical Engineering, University of Engineering and Technology, Lahore. His research interests include low-power computing, the Internet of Things, and embedded systems.

MUHAMMAD USMAN GHANI received the B.S. and M.S. degrees in computer science from the University of Engineering and Technology, Lahore, Pakistan, in 2010 and 2014, respectively. He joined the University of Engineering and Technology, Lahore, Pakistan, in 2020. He is currently serving as a Visiting Lecturer with Air University Islamabad. His major research topics include software engineering, artificial intelligence, and machine learning.

ATTA UR REHMAN received the B.Eng. degree in electronic engineering from Air University Islamabad, in 2006, and the M.Sc. (Hons.) and Ph.D. degrees from Loughborough University, U.K., in 2010 and 2014, respectively. He joined the University of Engineering and Technology, Lahore, in 2007, as a Lecturer. He worked as a Postdoctoral Research Associate with the Department of Automatic Control and Systems Engineering, The University of Sheffield, from December 2013 to January 2016. Since January 2016, he has been working as an Assistant Professor with the University of Engineering and Technology Lahore. His main areas of research include computer vision, machine learning, pattern recognition, and multi-target tracking.

ABDUL JALEEL received the B.S. degree in computer science and engineering and the M.S. and Ph.D. degrees in computer science from UET, in 2006, 2010, and 2019, respectively. He is currently working as an Assistant Professor with the Rachna College of Engineering and Technology, UET. His research interests include the development of self-managing software applications, the Internet of Things, edge computing, and machine learning.