Using Deep Reinforcement Learning to Automate Network Configurations for Internet of Vehicles

Xing Liu*, Cheng Qian†, Wei Yu†, David Griffith‡, Avi Gopstein‡ and Nada Golmie‡
*Sam Huston State University, USA
Email: xxl020@shsu.edu
†Towson University, USA
Emails: cqian1@students.towson.edu, wyu@towson.edu
‡National Institute of Standards and Technology (NIST), USA
Emails:{david.griffith, avi.gopstein, nada.golmie}@nist.gov

***Abstract*—In this paper, we address the issue of automating network configurations for dynamic network environments such as the Internet of Vehicles (IoV). Configuring network settings in IoV environments has proven difficult due to their dynamic and self-organizing nature. To address this issue, we propose a deep reinforcement learning-based approach to configure IoV network settings automatically. Specifically, we use a collection of neural networks to convert the observations of a communication environment (channel power gain, cross-channel power gain, etc.) into key features, which are then supplied to a deep Q neural network (DQN) as input for training. Afterward, the DQN will select the optimal network configuration for vehicles in the IoV environment. In addition, our approach considers both centralized and distributed training strategies. The centralized training strategy conducts the DQN training process on a roadside server, while the distributed training strategy trains the DQN on vehicles locally. Through our designed IoV simulation platform, we evaluate the efficacy of our proposed approach, demonstrating that it can improve the quality of services (QoS) in the IoV environments concerning reliability, latency, and service satisfaction.**

***Index Terms*—Internet of Vehicles, Deep Reinforcement Learning**

I. Introduction

Intelligent transportation systems (ITS) integrate network communication, computing, and sensor engineering technologies to achieve real-time automated transportation management [1]–[3]. According to Gartner's forecast, more than 740 000 autonomous-ready vehicles will be added to the market in 2023 [4]. These autonomous vehicles rely on developing supporting infrastructure (specialized computer hardware, intelligent software agents, and reliable communication channels). Internet of Vehicles (IoV) is a key network architecture that supports communication among smart-enabled vehicles and supporting devices (e.g., roadside units (RSUs)). The new era of IoV has several distinct characteristics, including large-scale support, high dynamicity, and strict quality of service (QoS) requirements (latency, reliability, autonomy, and efficiency, among others), which shall be considered in its design [5]. In dynamic IoV scenarios, the communication environment changes rapidly, presenting difficulties for packet scheduling tasks, especially concerning strict QoS requirements (e.g., acceptable delay, packet delivery rate).

Deep learning-based techniques, such as deep reinforcement learning, leverage deep neural networks (DNNs) to approximate state-action value functions for solving sequential decision-making problems in the Markov decision process (MDP) [6]. Deep reinforcement learning techniques have been widely adopted to solve complex problems in a variety of domains, such as the cooperative multi-agent control [7], communication and control of industrial Internet of Things (IIoT) [8], autonomous IoT, and robots [9], energy-efficient computational offloading [10], and autonomous driving [6]. For example, Ning *et al.* in [10] proposed a deep reinforcement learning framework to conduct computational offloading such that energy efficiency in IoV scenarios can be improved considering delay constraints.

To ensure the QoS of IoV in heterogeneous and dynamic networking environments, in this paper, we propose a deep reinforcement learning-based approach that assists with the automation of IoV configuration. In dynamic IoV environments, the communication links could be unreliable, significantly affecting QoS performance. To deal with this issue, we design a deep Q learning-based approach to reconfigure network settings so that QoS requirements are satisfied automatically. Specifically, our approach consists of three phases: (i) data preparation, (ii) deep Q neural network (DQN) training, and (iii) distribution of control. Based on how the DQN model is trained, we consider both centralized DQN and distributed DQN strategies. In the centralized DQN strategy, the IoV system collects data from IoV vehicles, which is used to train the DQN at the RSU server. Next, the DQN updates its parameters based on the collected data during training. Finally, the trained DQN is forwarded to the vehicles, which automatically assists vehicles in configuring their network settings. In the distributed DQN strategy, each IoV vehicle trains its own DQN model and shares the learned information with other vehicles in the network via learning aggregation to improve accuracy. An aggregated model is stored on the RSU server for other vehicles.

To summarize, we make the following contributions: (i) We address the QoS issues for heterogeneous and dynamic IoV network environments where communication links are unstable and unreliable. We propose two deep reinforcement learning-based strategies to automatically configure IoV net-

working settings such that QoS in different scenarios can be improved. (ii) We compare our proposed approach with two other IoV network configuration strategies (i.e., random selection and single DQN) in two classic IoV scenarios: urban and highway areas. We conduct extensive performance evaluations based on our designed IoV simulation platform. Our experimental results demonstrate that our proposed approach can significantly increase the performance of IoV in terms of reliability, latency, and service satisfaction.

The remainder of this paper is organized as follows: In Section II conducts a literature review of relevant studies on IoV communications. Section III presents the system model of IoV communications. Section IV describes our proposed approach with centralized and distributed model training strategies in detail. Section V describes the evaluation of validating the efficacy of our approach. Finally, Section VI summarizes the paper.

II. Related Work

IoV Network Architecture: The IoV is facilitated by supporting devices inside the vehicle (e.g., cameras, sensors) and structures outside the vehicle (e.g., buildings, RSUs). IoV devices are interconnected through heterogeneous communication systems such as sensor networks, global positioning systems (GPS), cellular networks, Bluetooth, radio frequency identification (RFID), Wi-Fi, etc. Richa *et al.* [11] studied heterogeneous-IoVs (HET-IoVs) and proposed the four-layer IoV architecture, including the environment sensing layer, heterogeneous networking layer, coordination control layer, and application service layer. Jiang *et al.* [12] proposed a measurement framework that characterizes IoV performance for IoV, including delay, loss, throughput, and others.

QoS-aware Network Management in IoV: The performance (i.e., QoS) of IoV, which can be characterized by transmission delay, throughput, packet loss, bitrate, jitter, and availability, directly affects the safety and security of IoV applications, such as traffic management, autonomous driving, and so on [13]. Hussain *et al.* [14] reviewed the QoS challenges and issues that must be addressed to meet users' needs and prevent any packet loss that could lead to catastrophic results in IoV scenarios. Hammoud *et al.* [15] proposed the deployment of fog servers in federated learning architecture to avoid QoS deterioration and secure IoV services. To improve the QoS and securing IoV, Xu *et al.* [16] utilized a real-time analysis mechanism based on the improved cuckoo search (ICS) algorithm to predict the outage probability (OP). Besides, Hou *et al.* [17] used an edge computing-enabled software-defined IoV (EC-SDIoV) to provide low-latency computing services through SDNs. Moreover, Zhai *et al.* [18] proposed an offloading model based on fog computing and SDNs to reduce the energy consumption of IoV.

Deep Reinforcement Learning in IoV: Due to the enormous size of IoV, automated management and configuration techniques, such as network intelligence, become the key enablers for efficient and intelligent IoV. Deep reinforcement learning techniques have been adopted for automatically configuring smart-enabled devices and improving QoS [19]. To be specific, Abbasi *et al.* [19] extensively surveyed deep reinforcement learning schemes in the Medium Access Control (MAC) layer to ensure QoS. Ning *et al.* [10] leveraged deep reinforcement learning to conduct computational offloading for energy efficiency considering QoS in intelligent IoV. Likewise, Sodhro *et al.* [20] proposed an artificial intelligence (AI)-enabled QoS optimization framework for IoV multimedia communications. Moreover, Li *et al.* [21] proposed an information-centric, network-based framework to integrate IoV services.

III. System Model

We consider an IoV communication network environment of K vehicles within the communication area. Each vehicle has a single antenna to communicate with other vehicles or the RSU. RSUs are equipped with antennas on top of base stations/access points deployed on the roadside. Also, RSUs are connected to a high-performance computing server, carrying out model training and analytics. We assume the network connection between the RSU and the server has sufficient network bandwidth to transmit all training data in time reliably. To ensure that the RSU collects data effectively without losing generality, we assume all the links between RSU and vehicles are assigned orthogonal radio resources.

Considering that vehicles share spectrum resources while conducting vehicle-to-vehicle (V2V) communications, we denote the received signal-to-interference-plus-noise-ratio (SINR) of vehicle k at channel m by $\gamma_k[m] = \frac{\rho_k[m]P_k h_k[m]}{I_k[m]+\sigma^2}$, where $\gamma_k[m]$ represents the SINR of vehicle k at channel m. The Boolean variable $\rho_k[m]$ indicates whether the channel m is chosen by vehicle k. If the vehicle k chooses channel m, we have $\rho_k[m] = 1$; otherwise, $\rho_k[m] = 0$. Denote P_k as the transmission power of vehicle k and $h_k[m]$ as the cross-channel power gain of vehicle k at channel m, respectively. Also, denote σ^2 as the noise power level of the current communication environment. Finally, $I_k[m]$, the interference power level received by vehicle k at channel m can be represented as, $I_k[m] = \sum_{l\neq k}^{K} \rho_l[m]P_l h_l[m]$, which is the cumulative interference of all other vehicles within the communication range of vehicle k on channel m.

We assume each vehicle has several communication tasks that must be completed within a time unit. Vehicles can freely choose the modulation type, data rate, and transmission spectrum to satisfy their tasks. In our case study, we assess the QoS for a task using two factors: end-to-end delay and packet delivery rate. End-to-end delay is the total time taken from communication task creation to completion or task discard. The packet delivery rate is defined as the ratio of successfully transmitted packets to the total number of packets sent. This definition applies to both our proposed centralized and distributed training strategies.

To quantify the performance of communication services in IoV, we design a QoS measure function as follows:

$$r_k^t = \begin{cases} \frac{w_d*(\tau_t - d_k^t)}{\tau_t} + \frac{w_p*(p_k^t - q_t)}{q_t}, & (d_k^t < \tau_t) \wedge (p_k^t > q_t) \\ 0 & (d_k^t > \tau_t) \vee (p_k^t < q_t) \end{cases} \tag{1}$$

Notations	Parameters
τ	Delay tolerance
γ	SINR
m	Communication channel number
Mo	Modulation type
Dr	Data rate
k	Vehicle ID
ρ	Channel Selection as boolean variable
σ	Noise power level
P	Transmission power
I	Interference power level
h	Cross channel power gain
R	Reward function
r	QoS measurement
t	Communication task
d	End-to-end delay
p	Packet delivery rate
w	Weight
S	State
Q	Q value
A	Action
θ	Neural network parameters
β	Contribution rate of local node

TABLE I. List of key notations

Here, τ_t is the delay tolerance of communication task t. The end-to-end delay of task t on vehicle k is represented by d_k^t. Also, q_t is the packet delivery rate tolerance of task t, and p_k^t is the packet delivery rate of task t on vehicle k. Furthermore, w_d is the weight that we assign for the end-to-end delay in the QoS evaluation function and w_p is the weight that we assign for the packet delivery rate in the QoS evaluation function.

We specifically consider vehicle communication and network configuration. Based on the self-organization of VANET and the speed of fast-moving vehicles, the network communication environment in IoV is very unstable and noisy. To solve this problem, the IEEE proposed the IEEE 802.11p protocol that supports vehicle-to-vehicle communication, and we configure the IoVs based on this protocol. In detail, based on the QoS, we optimize the selection of modulation type, data rate, and communication channel among the possible options for IEEE 802.11p. The QoS-based optimization problems for IoV network configuration is formulated by $\max \sum_k r_k^t(i,j,k)$, where $i \in M$ ($M = 1, 2, \ldots, m$), $j \in Mo$ ($Mo = 1, 2, \ldots, mo$, and $k \in Dr$ ($Dr = 1, 2, \ldots, dr$). Here, i is the available communication channel, j is the possible modulation type, and k is the achievable data rate.

IV. Our Approach

A. Problem Space

The problem of applying machine learning techniques to IoT is shown in Fig. 1. Here, the first dimension is the application space, including different IoT applications (smart transportation, smart grid, and smart city). The second dimension represents the three major categories of machine learning techniques (reinforcement learning, supervised learning, and unsupervised learning). The third dimension represents the model training in a centralized or distributed manner. Given the distributed nature of the IoV, RSUs are not always available for vehicle connectivity (i.e., RSU not in range). Thus, we should consider two scenarios: a centralized topology where the RSU acts as a communication and computing center. The other scenario is when no RSU is available, and the vehicle is connected in a decentralized manner (i.e., communicate via relay). To cover these two possible scenarios, we propose two deep reinforcement learning-based strategies to conduct model training for smart transportation applications. One uses a centralized strategy to train the DQN, while the other uses a distributed training strategy to train the DQN. Thus, we map our work to the shadow region of the problem space in Fig. 1. In the centralized training strategy, we train the DQN on an RSU in a central location. In the distributed training strategy, we train the DQN on vehicles locally.

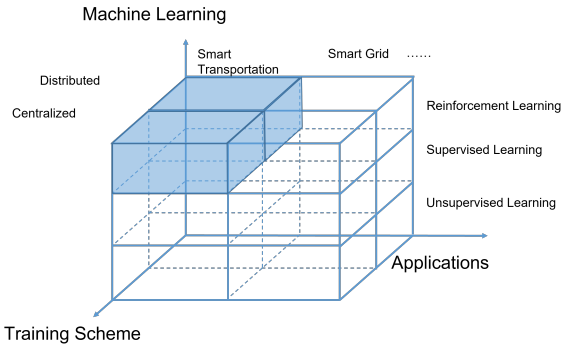

Fig. 1. Problem Space for applying machine learning to IoT applications

B. Centralized Training Strategy

As shown in Fig. 2, we leverage deep reinforcement learning and DNNs to configure the network settings in the IoV environment automatically. The strategy consists of three key phases: (i) **Data Preparation Phase:** The system generates and collects data that train the DQN. The RSU randomly assigns some tasks for vehicles in the covered area to generate the training data. The vehicles collect observations of the current communication environment and use a DNN to extract key features. Then, the key features extracted from different vehicles are aggregated by another DNN at the RSU server. The aggregated features are used as the training data for the DQN at RSU server. (ii) **Centralized DQN Training Phase:** The DQN at the RSU server updates the weights of the DQN and DNN during the training process. After several epochs of training, the RSU forwards the current DQN and DNN network parameters to all the vehicles in the covered area. The vehicles use these neural network parameters to generate more data. (iii) **Local Vehicle Control Phase:** The trained DQN is forwarded to all the vehicles in the scenario as a decision controller. The DQN then configures the network based on observing the current communication environment.

1) Data Preparation: We assume that the RSU randomly assigns different tasks to all the vehicles in the covered area. The vehicles take random actions affecting the modulation type, data rate, and selected channel to complete tasks. The purpose of this step is to generate diverse training data. Afterward, the vehicles record the local observation of the communication environment, including the transmission power,

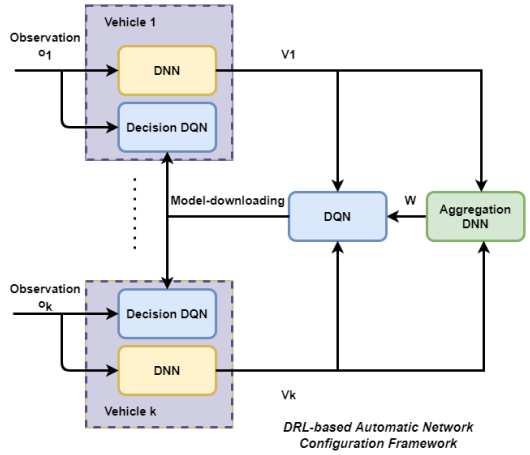

Fig. 2. Centralized Automatic IoV Network Configuration Framework

current channel power gain, and the cross-channel power gain of all channels. The k^{th} vehicle knows the transmission power (P_k) of itself. The k^{th} vehicle's current channel power gain ($h_k[m]$) and interference power from all other channels ($I_k[m]$) can be accurately estimated by the receiver vehicle and sent back to the transmitter vehicle. Thus, the local observation at k^{th} vehicle can be represented by the ordered triple: $o_k = \{P_k, h_k, I_k\}$.

Because raw data from observations might be large with many input values, transmitting raw data to the RSU is inefficient and increases the computational overhead of training DQN. To deal with this issue, as shown in Fig. 2, we use a DNN at each vehicle to extract the key features of the observations and only transmit the extracted features v_k to the RSU server for training the DQN. The extracted features v_k contain all relevant information, reflecting the current communication environment around the k^{th} vehicle. Another DNN on the RSU server aggregates extracted features from different vehicles to a central location. Thus, a complete estimation w of the current communication environment can be obtained by the aggregation DNN. If each vehicle only considers local observations, it will naturally choose actions to maximize the rewards of its tasks, which may cause other vehicles to take similar actions to compete for limited spectrum resources. As a result, the overall reward of the whole scenario will not be maximized. Nonetheless, by considering the whole communication environment, the DQN can use the set of K sets of extracted features to determine how to maximize the total rewards of all the vehicles in the IoV scenario.

2) Centralized DQN Training: In this phase, we train the DQN based on the framework in Fig. 2 by using the aggregated communication environment and extracted features of local observations. The DQN implements a reinforcement learning algorithm that aims to find the optimal actions to maximize the total QoS over the K vehicles. In reinforcement learning, the agent estimates the effectiveness of its actions by interacting with the environment. At each discrete time slot t, after the agent takes an action a_t, it drives the agent from the current state S_t to another state S_{t+1} with a transition probability p and obtain a reward R_{t+1}.

In our case, the state s of the environment is the overall combination of the communication environment w and the set of local observations $\{v_k\}_{k=1}^{K}$. Recall the data preparation phase, where the overall communication environment w is extracted from all the local observations v_k. Thus, the state s is expressed as $s = \{v_1, v_2, \ldots, v_K, w\}$. Then, the action a combines modulation type Mt, data rate Dr, and the selected channel Sc. All vehicles use the same action space since they inherit the same global model in the centralized training strategy. The action a is expressed as: $a = \{Mt_k, Dr_k, Sc_k\}_{k=1}^{K}$.

Finally, we design the reward function for our DQN algorithm, which is critical for the controller's performance and designed for fulfilling QoS requirements for different tasks in the IoV environment. Thus, the reward function is the cumulative reward of our proposed QoS measurement function for all vehicles in the designated scenario. The reward R can be represented as $R = \sum_{k=1}^{K}\sum_{t=1}^{T} r_k^t$, where r_k^t is the QoS measurement function that we mentioned in Section III. Two parts construct this measurement, the end-to-end delay tolerance of the communication and the packet delivery rate tolerance. The reward becomes zero if the current delay is greater than the maximum tolerance rate or the current packet delivery rate is less than the minimum. Otherwise, the weighted ratio of the current delay and the tolerated delay is added to the weighted ratio of the current packet delivery rate and the tolerance packet delivery rate. Also, K is the total number of vehicles in the investigated scenario, and T is the total number of tasks assigned to the vehicle.

The reinforcement learning problem is summarized as finding an optimal policy $\pi^*(a, s)$ that indicates the probability of choosing the action a in the action space A at state s. To solve the problem, we leverage the classic Q-learning scheme, which computes the value of an action (q value) via well-known Bellman equations. Here, the q value indicates the expected reward returned for taking action a at state s and the discounted cumulative reward for all subsequent states before reaching the end state.

Mathematically, the q value can be updated by using the following iterative equation, $Q(S[t], A[t]) \leftarrow Q(S[t], A[t]) + \alpha[R[t+1] + \gamma \underset{a}{max} Q(S[t+1], a) - Q(S[t], A[t])]$ [22], where α is the step-size parameter and γ is the discount factor. Also, to choose the near-optimal action a at state s, some exploration policies are leveraged to avoid the algorithm from falling into the local optimal point. One of the most widely used exploration policies is the ε-greedy policy. The ε-greedy policy chooses the action a, which has the maximum q value at state s with probability $1 - \varepsilon$. It chooses a random action at state s with probability ε, expressed by

$$A \leftarrow \begin{cases} \arg\max_a(s, a), & \textit{probability} \quad 1 - \varepsilon \\ \textit{random action}, & \textit{probability} \quad \varepsilon \end{cases} \tag{2}$$

Due to the exponential increase in computational complexity, the classic Q-learning algorithm can only work with limited state and action spaces. For large state and action spaces, a function should be used to approximate the q value, instead of storing all q values in a table. This can reduce the storage

space to store all q values and estimate the q value of an unseen state-action pair.

In our case, we adopt the DQN approach, which uses a DNN to estimate the q value of a state-action pair. The approximation function is represented by the network parameters θ of the DNN. The DQN also adopts ε-greedy policy to explore the state-action space. After an action is taken, the experience is stored in replay memory as a tuple $T = (S[t], A[t], R[t+], S[t+1])$. Within an individual time step, a mini batch of stored experiences is sampled in a uniform manner to train the DNN in the DQN, and update the network parameters θ. Algorithm 1 shows the procedure in Fig. 2 to update the network parameters.

Algorithm 1: Centralized DQN training

Input: DNN for each vehicle, Aggregation DNN, QoS-based reward
Output: DNN for each vehicle, Aggregation DNN, the optimal policy π^*, network parameters θ

Initialize all parameters in DNNs and DQN model;
while *episode* i< *maximum episode* I & t< *maximum simulation time* **do**
  i ++;
  Each vehicle collects local observations using its own DNN, and sends the extracted features $v_k[t]$ to the Aggregated DNN;
  The Aggregated DNN each $v_k[t]$ as input, and generates global observation w^t;
  The DQN takes $v_k[t]$ and $w[t]$ as input states;
  The DQN generates action $a_k[t]$ with ε-greedy policy;
  Each vehicle downloads the DQN model and executes the action;
  Each vehicle obtains the reward $R[t+1]$ and new observation $o_k[t+1]$;
  Save the experience tuple data $\{(o_k[t], w[t]), a[t], R[t+1], (o_k[t+1], w[t+1])\}$ into replay memory;
  Sample a mini-batch of data from replay memory;
  Use sampled data to train all DNNs and the DQN;
  Each vehicle updates the local DNN every n steps;
  t ++;
end

3) Local Vehicle Control Phase: After deploying well-trained DQN to the local vehicles. The k^{th} vehicle can use the local observation o_k and the extracted global features w as inputs for its decision. As our DQN is trained while considering the QoS of all vehicles within the covered area, the optimal policy π^* will maximize the cumulative QoS quantity function. This means that our DQN is trained to find a policy to satisfy the most QoS requirements in this scenario.

C. Distributed Training Strategy

As shown in Fig. 3, in the distributed IoV network configuration automation, we train the DQN on vehicles locally, consisting of three key phases: (i) *Data Preparation Phase*, (ii) *Distributed DQN Training Phase*, and (iii) *Local Vehicle Control Phase*. As the data preparation and local vehicle control phases are the same as what was discussed in Section IV-B, we focus on the distributed DQN training phase in detail.

1) Distributed Training Phase: We train the DQN based on the workflow in Fig. 3. The distributed strategy trains the DQN on each vehicle locally. Consequently, using only local observations to train the DQN leads to biased results as vehicles cannot collect the global channel information. To

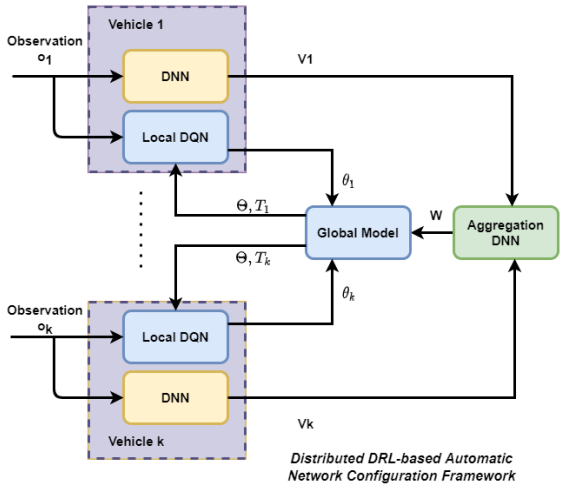

Fig. 3. Distributed IoV Network Configuration Automation

tackle this issue, we aggregate training models from different local vehicles to improve the versatility and accuracy of the DQN model.

Algorithm 2 describes the detailed procedure of realizing this distributed training strategy. First, we initialize a global model on the RSU, which, in addition to the sampled tuples T_k that are sent to the k^{th} vehicle, is transmitted to all vehicles in the covered region. Recall that T_k is used by the k^{th} vehicle but has the same structure as the tuple defined in Section IV-B2. The vehicles train this model with local observations and update their local models.

We propose a weighted updating method using the following policy to update the local DQN:

$$\theta_k^{'} = \omega\theta_k + (1-\omega)\Theta. \tag{3}$$

Here, ω is the update weight, θ_k is the set of local DQN model network parameters, and Θ is the global DQN model network parameters. The update weight ω lies in the unit interval, $[0, 1]$, representing the ratio of local network parameters to global network parameters in the update process. The local network parameters θ_k at the kth vehicle are produced by training the local DQN model with local observations and the tuple T_k. After updating the local DQN model for M steps, the local nodes upload the network parameters of their local DQN model to the RSU. The RSU updates the global DQN model by aggregating the local parameters from each local node by using

$$\Theta = \sum_{k \in K} \beta_k \theta_k. \tag{4}$$

Here, β_k is the local node k contribution rate to the global DQN model. The contribution rate of local node k can be derived via

$$\beta_k = \bar{r_k} / \sum_{i \in K} \bar{r_i}, \tag{5}$$

where $\bar{r_k}$ is the average reward of a local DQN model during M steps of training. The contribution rate is the ratio of the average reward of a local DQN model to the sum of the average rewards of all local DQN models in the designated scenario.

Then, the RSU broadcasts the new global DQN model and the new tuple to local nodes and repeats the training process. We add part of the global channel information to the local DQN model through this weighted update method by adopting a partial global model.

Algorithm 2: Distributed DQN training

Input: DNN for each vehicle, Aggregation DNN, QoS-based reward
Output: DNN for each vehicle, Aggregation DNN, the optimal policy π^*, network parameters θ

Initialize all parameters in DNNs and DQN model;
while *episode* i< *maximum episode* I & t< *maximum simulation time* **do**
  i ++;
  Each vehicle collects local observations using its own DNN, and sends the extracted features $v_k[t]$ to the Aggregated DNN;
  The Aggregated DNN takes each $v_k[t]$ as input, and generates global observation $w[t]$;
  if *episode* i % M==*0* **then**
    Global model sends global parameters Θ, random tuple T_k to local node k;
    Local node k updates the local DQN model using Equation (3);
    The local DQN generates action $a_k[t]$ with ε-greedy policy;
    Each vehicle obtains the reward $R[t+1]$ and new observation $o_k[t+1]$;
  else
    Local node k updates the local DQN model using a local gradient;
    The local DQN generates action $a_k[t]$ with ε-greedy policy;
    Each vehicle obtains the reward $R[t+1]$ and new observation $o_k[t+1]$;
  end
  Local nodes send local parameters θ to the global model;
  Global model is updated using Equation (4);
  Save the experience tuple data $\{(o_k[t], w[t]), a[t, R[t+1], (o_k[t+1], w[t+1])\}$ into replay memory;
  t ++;
end

D. Complexity Analysis

We use the basic DDQN model as our DQN architecture for the general complexity analysis. We construct our DQN model with full connect layers, and in DDQN architecture, we have two identical DQN models. The computational complexity of each training step for one DQN model is $O(\sum_{b=1}^{B} W_{b-1}W_b)$, where W_{b-1} is the neural size of the b-th layer, and B is the total number of FC layers. Since in DDQN, the target DQN only conducts the computation during the forward propagation and gets updated by duplicating the other DQN every few steps, the total computational complexity of our framework for one node is $O(3I\sum_{b=1}^{B} W_{b-1}W_b)$, where I is the total training steps for one node.

For the centralized training strategy, a DNN is trained to extract the local observations' key parameters, and an aggregation DNN is trained. Considering they are all using FC layers, similarly, the total computational complexity will be $O(3I\sum_{b=1}^{B} W_{b-1}W_b + 2\sum_{c=1}^{C} X_{c-1}X_c + 2\sum_{d=1}^{D} Y_{d-1}Y_d)$, where X_{c-1} is the neural size of the c-th layer in local DNN, and Y_{d-1} is the neural size of d-th layer in aggregation DNN. For the distributed training strategy, as local nodes need to conduct the training for DNN and local DQN, the computational complexity is $O(3I\sum_{b=1}^{B} W_{b-1}W_b + 2\sum_{c=1}^{C} X_{c-1}X_c)$.

V. Performance Evaluation

A. Methodology

We evaluate our proposed approach in two scenarios: (i) *urban scenario* and (ii) *highway scenario*. In the urban scenario, we limit the maximum speed of vehicles to 8.9 m/s (20 miles per hour). In the highway scenario, we limit the maximum speed of vehicles at 29 m/s (65 miles per hour). We consider a 300 m $\times$ 300 m area with a maximum height of 50 m as the simulation area for each scenario. The simulation ran on a PC using the Windows 10 OS, 32 GB of memory, and an AMD 2700 CPU[1]. We used the simulation platform we proposed in our prior work [23], combining OMNET++ network simulator and SUMO traffic simulator. An Extensible Markup Language (XML) file defines any moving object's behavior in the IoT scenario. SUMO simulates the traffic flow, containing source location, destination location, vehicle speed, vehicle type, drivers' behavior, etc. of mobile objects. The source and destination are randomly generated. The traffic model is the default vehicle following model in SUMO.

We use IEEE 802.11p [24] as our communication protocol for all the vehicles in the simulated network. The total number of vehicles in the simulation varies from 5 to 50 vehicles, in increments of 5 vehicles. There is one RSU inside the simulation area. We use a 10 Mhz bandwidth with a carrier frequency of 5.9 Ghz as recommended in IEEE 802.11p. The noise level is set to -114 dBm (dBm is power level expressed in decibels (dB) regarding one milliwatt (mW)). We set the maximum end-to-end delay tolerance τ_t to 90 ms, meaning that a task is not completed within 90 ms, it is considered a failure. The vehicles can choose 4 different modulation types and 4 different data rates. The modulation type and data rate combinations follow the standard recommendations in IEEE 802.11p. The modulation types and data rates, and additional simulation parameters are shown in Table II. The action space we use in this paper is all combinations of possible modulation types, data rates, and available channels. Recall from Section IV-B2 that the reward function that we use in this simulation is $R = \sum_{k=1}^{K}\sum_{t=1}^{T} r_k^t$, where r_k^t is the QoS measurement function that we mentioned in Section III. Also, K is the total number of vehicles in the simulation scenario, ranging from 5 to 50. T is the total number of tasks assigned to the vehicle. We set 10 tasks for each vehicle, meaning T varies from 50 to 500.

The specific architecture of the DNN and DQN are summarized in Table III. The number of inputs for the local DNN is 3, corresponding to power, channel power gain, and cross-channel power gain. We use Fully Connected (FC) layers as our hidden layers. As there is no theoretical basis to find the optimal number of hidden layers and neurons quickly,

[1]Certain commercial equipment, instruments, or materials are identified in this paper to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

TABLE II. Simulation Parameters

Parameter	Value
Number of Users	5,10,15,20,25,30,35,40,45,50
Number of RSU	1
Carrier frequency	5.9 GHz
Bandwidth	10 MHz
Number of channels	10
Transmit Power	10 mW per 1 MHz
Noise power	-114 dBm
Time constraint of IoV transmission	90 ms
Payload size	[1,2,...] x 1060 bytes
Vehicle speeds	8.9 m/s, 29 m/s
Vehicle following model	Urban, Highway
RSU antenna height	25 m
User antenna height	1.5 m
Modulation type	BPSK, QPSK, 16-QAM, 64-QAM
Data rate	3 Mbit/s, 6 Mbit/s, 12 Mbit/s, 27 Mbit/s

we use parameter-tuning mechanisms to determine the neural network's structure. We chose these parameters by running multiple simulation trials with different values in each hidden layer. During the tuning phase, we try the range of hidden layers from 3 to 6. We run each simulation 20 times to determine our scenario's optimal number of hidden layers. Also, in each simulation run, we ran 3500 epochs. Considering that the aggregated DNN takes all the key features extracted from the local DNN as input, the number of inputs for the aggregated DNN is assumed to equal the sum of all the output quantities of each local DNN. However, the DQN uses data that contains the local key features and aggregated global key features for training. Thus, the number of inputs for the DQN should be the number of outputs of aggregated DNN plus the number of outputs of a local DNN.

We define the two baseline approaches for performance comparison: (i) *Random selection approach* where each vehicle randomly configures the IoV network settings without considering the current communication environment, and (ii) *Single DQN Approach:* that trains a local control DQN for each vehicle in the IoV scenario, only using local observation key features as input.

We investigate the performance of our proposed DQN approach from two perspectives: (i) *communication* that represents how well our proposed approach addresses the automatic network configuration problem, and (ii) *learning effectiveness* that shows how long it takes the DQN to complete the training process. From the communication perspective, considering our proposed approach focuses on improving QoS of the entire IoV scenario, we employ three metrics: (i) *packet delivery rate* computed by dividing the number of successfully received packets by the total number of packets sent, (ii) *end-to-end delay* referring to the time taken for a packet to be sent from the transmitter to the receiver, and (iii) *task satisfaction rate* that is a qualitative measurement for identifying communication service satisfaction, to compare the performance between the random action, the single DQN, and our proposed DQN approach. We estimate this by dividing the number of successfully completed tasks by the total number of assigned communication tasks. Each communication task has a different packet delivery rate and delay requirement. Only when both requirements are fulfilled, the task is considered satisfied; otherwise, the task is viewed as a failure.

We use the loss function to indicate how the DQN converges during training. According to the q value update method that we state in Section IV-B2, the loss can be computed by $L(\theta) = [R_{t+1} + \gamma \underset{a}{max} Q(S_{t+1}, a; \theta^-) - Q(S_t, A_t; \theta)]^2$, where θ is the set of the network parameters of the training q neural network. Within every few time steps, parameters are frozen to avoid short-term oscillations. Another target q neural network will duplicate the network parameters of the training q neural network and store them as the parameter set θ^-. By minimizing the square error of the q value estimated by these two deep neural networks using the stochastic gradient descent method, we can obtain a convergent DQN after sufficient training epochs.

TABLE III. DNN and DQN architecture

	DNN	Aggregated DNN	DQN
Input layer	4	K*N_d	N_q+N_d
Hidden layers	3 FC layers (16,32,16)	3 FC layers (800,500,300)	3 FC layers (120,240,120)
Output layer	N_d	N_q	160

B. Results

1) Centralized Training: **Urban Scenario Performance:** Fig. 4 illustrates the performance of the packet delivery rate using the two DQN approaches and the random baseline as the number of users increases in the urban scenario. When there are only five users in the IoV scenario, all three approaches can achieve a high packet delivery rate, and both DQN approaches achieve close to 100 % packet delivery rate when up to $K = 15$ vehicles are present. This is because, under such a situation, there are enough spectrum resources for all the users to complete their communication tasks. The packet delivery rate of the random selection approach begins to decrease when there are 10 users present and thus is lower than the packet delivery rates of both DQN schemes. This is because the random selection approach cannot avoid the packet collision problem when the number of users increases. Packet collision occurs when two packets are transmitted simultaneously on the same channel. The single DQN approach can handle the collision problem better than the random approach over the range of values for the number of users. Because the single DQN approach leverages local observation to make the decision, it chooses the available channel with the largest bandwidth, only maximizing the QoS score for its own tasks. When the number of users increases, the density of vehicles also increases. As a single DQN is only trained through local observations, vehicles compete for spectrum resources. When vehicles choose to transmit the packets on the same channel, some packets will be dropped due to high collision probability and SINR.

Our proposed DQN approach outperforms two other baseline approaches. The main reason is that our DQN approach considers the QoS of the entire IoV scenario, and our DQN is trained using the global communication key factor w. When there are less than 15 users, our proposed DQN approach achieves a near-perfect packet delivery rate. Even when the number of users increases to 30, our proposed DQN approach can maintain a packet delivery rate of about 80 %, while the other two approaches drop to about 50 %. When there are

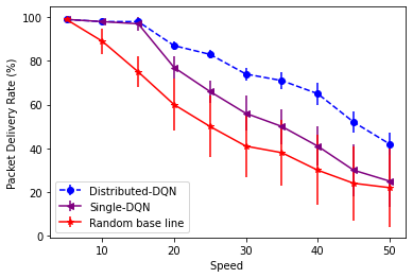

Fig. 4. Packet Delivery Rate of Centralized DQN in the Urban Scenario

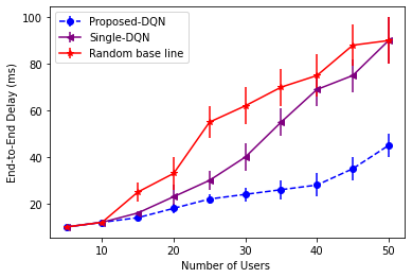

Fig. 5. End-to-End Delay of Centralized DQN in the Urban Scenario

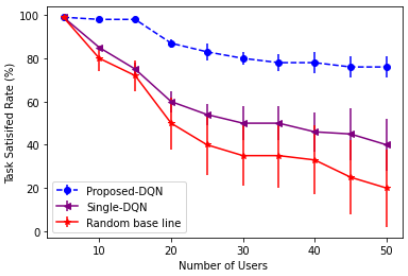

Fig. 6. Task Satisfied Rate of Centralized DQN in the Urban Scenario

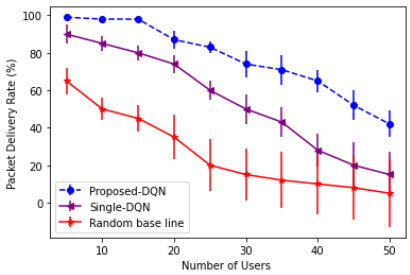

Fig. 7. Packet Delivery Rate of Centralized DQN in the Highway Scenario

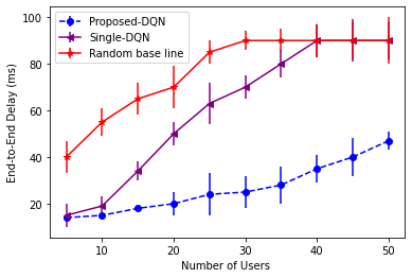

Fig. 8. End-to-End Delay of Centralized DQN in the Highway Scenario

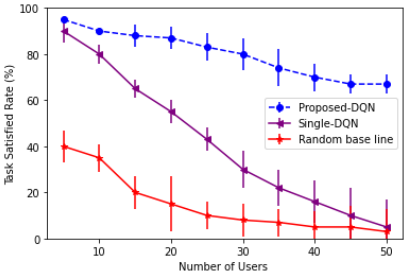

Fig. 9. Task Satisfied Rate of Centralized DQN in the Highway Scenario

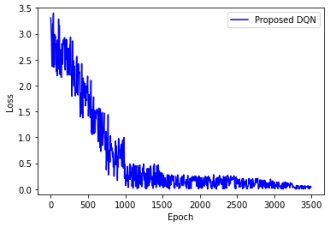

Fig. 10. Loss of Centralized DQN Approach in the Urban Scenario

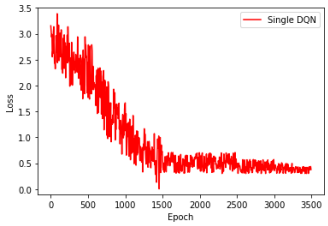

Fig. 11. Loss of Single DQN Approach in the Urban Scenario

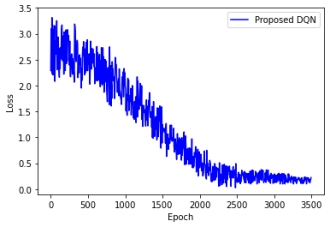

Fig. 12. Loss of Centralized DQN Approach in the Highway Scenario

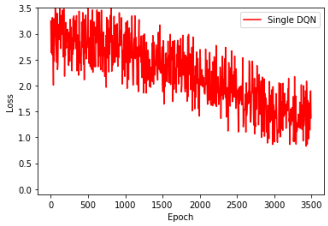

Fig. 13. Loss of Single DQN Approach in the Highway Scenario

many users, the packet delivery rate of our proposed DQN approach also drops. This is because there are not enough network resources to support all communication tasks in such circumstances.

Fig. 5 represents the end-to-end delay performance for three approaches when the number of users increases. Even when there are 50 users, our proposed DQN approach can keep the end-to-end latency around 50 ms. Fig. 6 shows the task satisfaction rate for all approaches when the number of users increases. Recall that our proposed DQN approach is trained to maximize task satisfaction to be superior to the other two approaches concerning satisfaction rate. By combining the results from Fig. 4 and Fig. 6, it can be seen that our proposed DQN approach intentionally drops some packets to maximize the task satisfaction rate. This is because we use the cumulative QoS function of all users within the covered area as the reward function to train DQN.

Figs. 10 and 11 show the change of DQN loss vs. the training epochs when the number of users is 5 and the speed of vehicles is limited to 8.9 m/s (25 miles per hour). This result confirms the convergence of the DQN and the time taken to reach the state of convergence. As we can see in comparing Fig. 10 to Fig. 11, compared to the single DQN, our proposed DQN approach has a smaller loss, a faster convergence process, and experiences smaller fluctuations during training. This is because our proposed DQN approach is trained with data from different vehicles. In contrast, a single DQN approach only relies on the data it generates, and it is difficult for a DQN trained with partial information to learn about the surrounding environment correctly. Thus, the overall performance of the single DQN approach is worse than that of our proposed DQN approach.

Highway Scenario Performance: Fig. 7 shows the performance of the packet delivery rate as the number of users increases in the highway scenario using the centralized DQN approach. In the highway scenario, the communication environment changes more rapidly, increasing communication difficulty in the IoV scenario. As seen in Fig. 8, all three approaches in the highway scenario led to a lower packet delivery rate than those in the urban scenario. When there are 5 users, the packet delivery rate of the random selection approach drops to around 65 %, and the packet delivery rate of the single DQN approach drops to around 90 %. At the same time, our proposed DQN approach only drops 2 % for

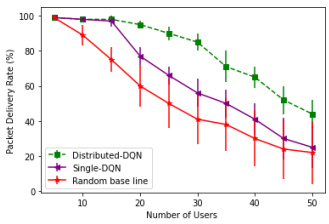

Fig. 14. Packet Delivery Rate of Distributed DQN in the Urban Scenario

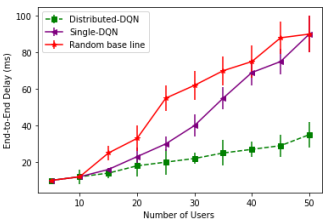

Fig. 15. End-to-End Delay of Distributed DQN in the Urban Scenario

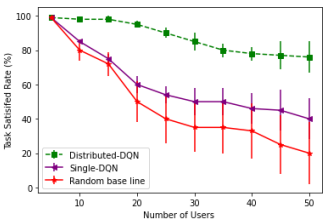

Fig. 16. Task Satisfied Rate of Distributed DQN in the Urban Scenario

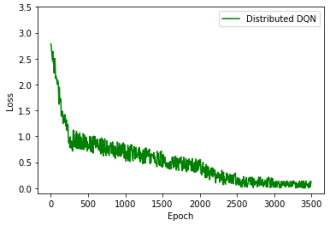

Fig. 17. Loss of Distributed DQN Approach in the Urban Scenario

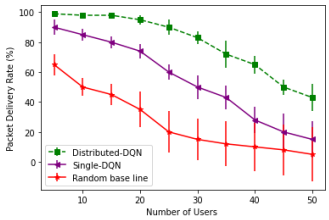

Fig. 18. Packet Delivery Rate of Distributed DQN in the Highway Scenario

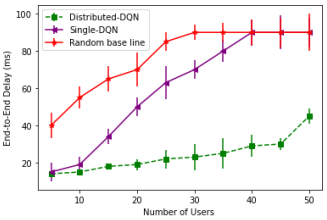

Fig. 19. End-to-End Delay of Distributed DQN in the Highway Scenario

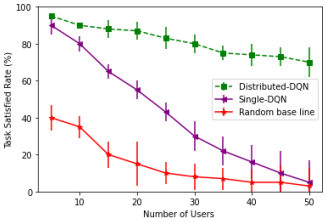

Fig. 20. Task Satisfied Rate of Distributed DQN in the Highway Scenario

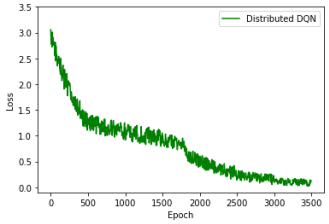

Fig. 21. Loss of Distributed DQN Approach in the Highway Scenario

the packet delivery rate.

Fig. 8 illustrates the end-to-end delay performance for the centralized DQN approach and the two baseline approaches in the highway scenario. The random selection approach reaches the 90 ms tolerance limit at 25 users, and the single DQN approach reaches the tolerance limit at 40 users. In contrast, our proposed approach can keep the end-to-end delay at 50 ms when there are 50 users in the IoV scenario. Finally, as shown in Fig. 9, our proposed DQN approach drops around 20 % for the task satisfaction rate with 50 users.

Figs. 12 and 13 show the change of DQN loss as the training epochs increase when there are 50 users and the vehicle speed is limited to 29 m/s (65 miles per hour). Comparing Fig. 10 to Fig. 12, we can see a steady drop in the loss until 2500 training epochs for the proposed centralized DQN approach, with convergence behavior persisting after 3500 training epochs. In contrast, after 3500 training epochs, the single DQN approach fails to achieve convergence. This is because the volatility of the environment hinders the DQN's capacity to estimate the transition probability with only partial information, as we discussed in Section IV-B2.

2) Distributed Training: **Urban Scenario Performance:** Fig. 14 shows the performance of the packet delivery rate as the number of users increases in the urban scenario. Our approach with the distributed training strategy outperforms the random selection approach and the single DQN approach. The performance of the distributed training strategy is slightly better than the centralized training strategy.

Fig. 15 illustrates the performance of the end-to-end delay for all approaches in the urban scenario. With fewer than 20 users, centralized and distributed DQN approaches have similar average delays. When there are more than 20 users, our distributed DQN approach has a smaller delay than the centralized DQN approach. Specifically, when there are 25 users, the distributed DQN approach has an average delay of 2 ms less than the centralized DQN. This gap gradually increases to about 10 ms as the number of users grows. Also, Fig. 16 illustrates the average task satisfaction rate of our distributed DQN approach in the urban scenario. When there are between 20 and 45 users, our distributed DQN has a higher average task satisfaction rate than the centralized DQN. With 50 users, it achieves almost the same task satisfaction rate as the centralized DQN results shown in Fig. 7.

Fig. 17 shows the result of training and the evolution of DQN loss for the distributed DQN approach in the urban scenario. Compared with the centralized DQN, the loss of the distributed DQN drops faster in the early stages of training. Around 250 training epochs, the loss of the distributed DQN approach drops more slowly, but the final average loss of the distributed DQN is slightly smaller than that of the centralized DQN. This is because the distributed DQN has multiple local nodes to train the DQN simultaneously. Thus, there is an advantage in the early exploratory stage of the training process. When the loss drops to a certain level, because each local node's training set is different in updating the global model, the gradient drops in different directions, canceling parts out. As the distributed strategy uses more machines to train the DQN, the finalized DQN model is more accurate.

Highway Scenario Performance: Fig. 18 shows the performance of the packet delivery rate as the number of users increases in the highway scenario when we use the distributed DQN. Unlike the centralized DQN, in a volatile environment, the distributed DQN does not significantly reduce the performance of the packet delivery rate. This is because the simultaneous training of multiple local DQN models makes the global model more versatile initially, so it can quickly stabilize even in a volatile environment. Compared with the results in Fig. 14, the packet delivery rate only drops 1 % when the number of users reaches 50.

Fig. 19 illustrates the end-to-end delay performance of our

approach, with the distributed DQN and the two baseline approaches in highway scenarios. Our approach with the distributed DQN can keep the end-to-end delay at 45 ms, which is 5 ms lower than the centralized DQN with 50 users in the IoV scenario. As shown in Fig. 20, with 50 users, the average task satisfaction rate of the distributed DQN in the highway scenario is about 6 % lower than that in the urban scenario. Nonetheless, in the highway scenario, the average task satisfaction of the distributed DQN is still 3 % higher than the centralized DQN. Fig. 21 shows the training evolution of DQN loss for the distributed DQN in the highway scenario. Due to rapid environment changes in the highway, the loss value of the DQN model drops slower compared to the urban scenario. Nonetheless, the distributed DQN can still reach convergence after 3500 training epochs.

VI. Final Remarks

This paper proposed a deep reinforcement-based approach with centralized and distributed training strategies for automating IoV network configurations. The centralized training strategy suits road sections with considerable server computing power, such as those covered by RSU. The distributed training strategy is suitable for road sections where RSU cannot be selected and only surrounding vehicles are available. In the centralized DQN strategy, we used local DNNs on vehicles to convert local observations into key communication features. Next, the aggregation DNN on the RSU server takes all the key features from different vehicles and aggregates them into a global communication environment. Afterward, the RSU server trains the control DQN using the global communication environment and local observation features. Finally, all vehicles in the communication area download the trained DQN as the network configuration controller. In the distributed DQN strategy, we designed a weighted update method to aggregate the local DQN models trained by each IoV vehicle, improving the DQN models' accuracy. Based on local observations and the global communication environment, the control DQN automatically generates the optimal network configurations, considering the QoS for the entire IoV scenario. The evaluation results show that the centralized DQN and distributed DQN strategies have desirable performance concerning packet delivery rate, end-to-end delay, and task satisfaction rate on two classical IoV scenarios.

References

[1] M. B. Mollah, J. Zhao, D. Niyato, Y. L. Guan, C. Yuen, S. Sun, K.-Y. Lam, and L. H. Koh, "Blockchain for the internet of vehicles towards intelligent transportation systems: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4157–4185, 2020.

[2] N. Ekedebe, C. Lu, and W. Yu, "Towards experimental evaluation of intelligent transportation system safety and traffic efficiency," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 3757–3762.

[3] X. Liu, C. Qian, W. G. Hatcher, H. Xu, W. Liao, and W. Yu, "Secure internet of things (IoT)-based smart-world critical infrastructures: Survey, case study and research opportunities," *IEEE Access*, vol. 7, pp. 79 523–79 544, 2019.

[4] M. Rimol. Gartner forecasts more than 740,000 autonomous-ready vehicles to be added to global market in 2023. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-11-14-gartner-forecasts-more-than-740000-autonomous-ready-vehicles-to-be-added-to-global-market-in-2023

[5] K. Liu, X. Xu, M. Chen, B. Liu, L. Wu, and V. C. Lee, "A hierarchical architecture for the future internet of vehicles," *IEEE Communications Magazine*, vol. 57, no. 7, pp. 41–47, 2019.

[6] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[7] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

[8] H. Xu, X. Liu, W. G. Hatcher, G. Xu, W. Liao, and W. Yu, "Priority-aware reinforcement-learning-based integrated design of networking and control for industrial internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4668–4680, 2021.

[9] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Towards deep Q-network based resource allocation in industrial internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[10] Z. Ning, P. Dong, X. Wang, L. Guo, J. J. Rodrigues, X. Kong, J. Huang, and R. Y. Kwok, "Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1060–1072, 2019.

[11] T. Sharma, A. K. Sharma *et al.*, "Heterogeneous-internet of vehicles (Het-IoV) in twenty-first century: a comprehensive study," in *Handbook of Computer Networks and Cyber Security*. Springer, 2020, pp. 555–584.

[12] D. Jiang, Z. Wang, L. Huo, and S. Xie, "A performance measurement and analysis method for software-defined networking of IoV," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3707–3719, 2020.

[13] U. Z. A. Hamid, H. Zamzuri, and D. K. Limbu, "Internet of vehicle (IoV) applications in expediting the implementation of smart highway of autonomous vehicle: A survey," in *Performability in Internet of Things*. Springer, 2019, pp. 137–157.

[14] S. A. Hussain, K. M. Yusof, S. M. Hussain, and A. V. Singh, "A review of quality of service issues in internet of vehicles (iov)," in *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 380–383.

[15] A. Hammoud, H. Otrok, A. Mourad, and Z. Dziong, "On demand fog federations for horizontal federated learning in iov," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3062–3075, 2022.

[16] L. Xu, H. Wang, and T. A. Gulliver, "Outage probability performance analysis and prediction for mobile IoV networks based on ICS-BP neural network," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3524–3533, 2020.

[17] X. Hou, Z. Ren, J. Wang, W. Cheng, Y. Ren, K.-C. Chen, and H. Zhang, "Reliable computation offloading for edge-computing-enabled software-defined IoV," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7097–7111, 2020.

[18] Y. Zhai, W. Sun, J. Wu, L. Zhu, J. Shen, X. Du, and M. Guizani, "An energy aware offloading scheme for interdependent applications in software-defined IoV with fog computing architecture," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3813–3823, 2021.

[19] M. Abbasi, A. Shahraki, M. J. Piran, and A. Taherkordi, "Deep reinforcement learning for QoS provisioning at the MAC layer: A survey," *Engineering Applications of Artificial Intelligence*, vol. 102, p. 104234, 2021.

[20] A. H. Sodhro, Z. Luo, G. H. Sodhro, M. Muzamal, J. J. Rodrigues, and V. H. C. De Albuquerque, "Artificial intelligence based QoS optimization for multimedia communication in IoV systems," *Future Generation Computer Systems*, vol. 95, pp. 667–680, 2019.

[21] J. Li, J. Tang, J. Li, and F. Zou, "Deep reinforcement learning for intelligent computing and content edge service in ICN-based IoV," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–7.

[22] M. A. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, no. 3, p. 729, 2012.

[23] X. Liu, W. Yu, C. Qian, D. Griffith, and N. Golmie, "Integrated simulation platform for internet of vehicles," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 2756–2761.

[24] D. Jiang and L. Delgrossi, "IEEE 802.11p: Towards an international standard for wireless access in vehicular environments," in *VTC Spring 2008-IEEE Vehicular Technology Conference*. IEEE, 2008, pp. 2036–2040.