DSORL: Data Source Optimization With Reinforcement Learning Scheme for Vehicular Named Data Networks

Daniel Mawunyo Doe, *Member, IEEE*, Dawei Chen, *Member, IEEE*, Kyungtae Han, *Senior Member, IEEE*, Haoxin Wang, *Member, IEEE*, Jiang Xie, *Fellow, IEEE*, and Zhu Han, *Fellow, IEEE*

***Abstract*— Highly-dynamic (HD) map is an indispensable building block in the future of autonomous driving, allowing for fine-grained environmental awareness, precise localization, and route planning. However, since HD maps include rich, multidimensional information, the volume of HD map data is substantial and cannot be transmitted frequently by several vehicles over vehicular networks in real-time. Therefore, in this paper, we propose a data source selection scheme for effective HD map transmissions in vehicular named data networking (NDN) scenarios. To achieve our goal, we created a vehicular NDN environment for data collection, processing, and transmission using the CARLA simulator and robot operating system 2 (ROS2). Next, due to our vehicular NDN's dynamic and complex nature, we formulate the data source selection problem as a Markov decision process (MDP) and solve it using a reinforcement learning approach. For simplicity, we termed our proposed scheme data source optimization with reinforcement learning (DSORL), which selects suitable vehicles for HD map data transmission to MEC servers. The experiment results indicate that our suggested method outperformed existing baseline schemes, such as RLSS, Pro-RTT, and HDM-RTT, across all performance criteria in the evaluation. For instance, the system throughput increases by 65% – 72.68% compared to other baseline systems. Similarly, the proposed approach can minimize packet loss rate, data size, and transmission time by up to 60.6%, 77.5%, and 54.1%, respectively.**

***Index Terms*— Highly-dynamic (HD) map, map data transmission, named data networking, reinforcement learning, and vehicular networks.**

Manuscript received 11 October 2022; revised 15 February 2023 and 12 May 2023; accepted 17 May 2023. Date of publication 1 August 2023; date of current version 4 October 2023. This work was supported in part by the Funds from Toyota Motor North America, Amazon; and in part by the U.S. National Science Foundation (NSF) under Grant 1910667, Grant 1910891, Grant 2025284, Grant CNS-2107216, Grant CNS-2128368, and Grant CMMI-2222810. The Associate Editor for this article was G. Mao. *(Corresponding author: Daniel Mawunyo Doe.)*

Daniel Mawunyo Doe is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA (e-mail: dmdoe@uh.edu).

Dawei Chen and Kyungtae Han are with the InfoTech Laboratories, Toyota Motor North America Research and Development, Mountain View, CA 94043 USA (e-mail: dawei.chen1@toyota.com; kyungtae.han@toyota.com).

Haoxin Wang is with the Department of Computer Science, Georgia State University, Atlanta, GA 30302 USA (e-mail: haoxinwang@gsu.edu).

Jiang Xie is with the Department of Electrical and Computer Engineering, University of North Carolina at Charlotte, Charlotte, NC 28223 USA (e-mail: jxie1@uncc.edu).

Zhu Han is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 446-701, South Korea (e-mail: zhan2@uh.edu; hanzhu22@gmail.com).

Digital Object Identifier 10.1109/TITS.2023.3292033

I. Introduction

A. Background and Motivation

WITH the fast development of mobile communications, vehicular sensing technologies, and autonomous driving, the internet of autonomous vehicles (AVs) has become a prevalent topic [1]. Recent autonomous vehicles can determine their accurate locations and construct collision-free routes using Highly-dynamic (HD) maps. HD maps offer more dependable sensing capability and assistance for the decision-making layer of autonomous driving, where latency is critical. HD maps are conceptual maps with three layers: 1) the road model layer, 2) the lane model layer, and 3) the localization model layer [2]. The road model is utilized for navigation planning, while the lane model is used for route planning based on current road and traffic circumstances. The localization model is used to find the car on the map, and the lane model can only help with vehicle perception if the vehicle is properly found on the map. The HD map is necessary for autonomous driving, but its amount of data is enormous compared to a typical electronic map. As a result, generating, transmitting, and storing the full HD map data onboard in real-time with minimal latency and high reliability is impracticable in vehicular networks.

Named data networking (NDN) is a prospective future networking architecture in which each piece of content is considered to be an entity in the network, which can overcome the shortcomings of the current host-based network architecture (i.e., TCP/IP) in the existing vehicular networks [3], [4]. NDN offers significant promise for the automotive network, such as facilitating vehicle mobility, data sharing, data naming, and a naming-based route forwarding approach [5], [6]. However, several technological obstacles exist to creating an efficient HD map update via construction and distribution strategy in the vehicle NDN scenario. First, existing approaches choose data sources via a communication model (vehicle-to-infrastructure (V2I) or vehicle-to-vehicle (V2V)) [7], [8], where throughput can be drastically decreased as the number of vehicles grows. Second, existing approaches choose data sources based on the round-trip time (RTT) between the data source and the vehicle. In this circumstance, vehicle status changes in real-time, particularly in complicated moving settings. As a result, the RTT measure cannot ensure the optimal selection outcome since other forms of the vehicle information (e.g., data size,

speed, and direction) are not taken into account. Finally, because of vehicle mobility, frequent data source handovers will result in frequent RTT modifications, and wasteful data transfers [9].

Vehicular wireless communication, processing, and caching capabilities have recently advanced significantly [10], [11]. As a result, HD map update work may be divided into multiple subtasks and processed via vehicular distributed computing [12]. First, idle computer resources in automobiles are completely exploited, which may boost resource utilization and cloud server performance. Second, in vehicular NDN, utilizing vehicles as data collectors and processors may minimize the transfer of substantial raw environmental data while improving overall system latency [13]. Extensive studies have been conducted on distributed computing for vehicular networks [14]. For example, [14], [15] collaborated to optimize input data movement and job allocation in wired data centers (DCs) to concurrently decrease inter-DC traffic, enhance throughput, and minimize delays. Furthermore, in wireless sensor networks, reducers and route selection are optimized in tandem to lower transmission costs [16]. However, since the input data for the HD Map update is gathered in real-time, the data localization difficulties in [15] are not practical when crowd-sensing is used.

Crowdsourced data has recently received a lot of attention for HD map updates, as shown in [17] and [18]. Crowdsourced data is road observation data acquired by low-cost crowdsourcing devices, which commonly contain a low-cost camera and a global navigation satellite system (GNSS) sensor [19]. Crowdsourcing devices are mounted on cars that traverse the same routes regularly, making a vast quantity of environmental data freely accessible. The main disadvantage of crowdsourcing data is its considerable uncertainty, particularly in complicated moving circumstances [5]. Furthermore, as the vehicle population increases, the throughput of crowdsourced data drastically decreases due to excessive updates [7]. From [20], due to the imbalance in the dataset, [21] proposed an HD-Map-guided rapidly-exploring random tree (HDM-RRT) by combining an HD map and a sampling-based approach to quickly obtain high-quality and feasible map updates in complex campus scenarios. However, the authors failed to explore optimal data source selection schemes in their proposed scheme. As a result, [22] suggested a reinforcement learning-based data source selection method (RLSS) for selecting HD map data sources in vehicular NDN. However, the authors only addressed data source selection for the HD map distribution process and not the HD map construction process. Therefore, deciding on a suitable vehicle to transmit the necessary data for HD map construction in vehicular NDN environments is an open and relevant problem.

B. Contributions

In this paper, we present a smart data source selection scheme for HD map data transmission in vehicular NDN, called data source optimization with reinforcement learning (DSORL). The scheme leverages reinforcement learning (RL) to decide which vehicles should transmit data to the multi-access edge computing server. We formulate the selection problem as a Markov decision process and solve it with deep reinforcement learning (DRL), specifically, the deep deterministic policy gradient (DDPG) algorithm. The DSORL framework consists of four key components: state space, action, policy, and reward to develop a selection policy. To capture the dynamics of vehicular scenarios, we use factors such as round-trip time, vehicle speed, driving direction, and information entropy to represent the state of a data source. Our reward function evaluates the performance of the selected data source based on transmission throughput, data size, and duration. The reward function evaluates the transmission performance based on throughput, data size, and duration time. To run DSORL, we simulate a vehicular NDN environment with the CARLA simulator and robotic operating system 2 (ROS2), then employ a crowd-sensing paradigm to continuously collect environmental data using AV sensors in our environment. We perform extensive simulations to validate the performance gains achieved by the DSORL scheme. In particular, the system throughput can increase by $65\% - 72.68\%$ compared with other baseline schemes. Also, the proposed scheme can reduce packet loss rate, transmission data size, and transmission time by up to 60.6%, 77.5%, and 54.1%, respectively.

The major contributions of this work are summarized as follows.

- Our research is one of the first to investigate the use of a reinforcement learning-based strategy to HD map data source selection in vehicle NDN scenarios. We propose the DSORL framework for selecting the appropriate data source for transmission to MEC servers.
- We formulate the data source selection problem as a Markov decision process and use a DRL-based approach to solve it, specifically the DDPG algorithm. To optimize the selection's performance, we design a reward function that takes into account various measurements of data sources under vehicular NDN conditions.
- We perform extensive simulations to validate the performance gains achieved by the DSORL scheme. In particular, the system throughput can increase by $65\% - 72.68\%$ compared with other baseline schemes. Also, the proposed scheme can reduce packet loss rate, transmission data size, and transmission time by up to 60.6%, 77.5%, and 54.1%, respectively.

The following is an overview of the paper's structure. The system model is described in Section II. Section III presents DRL-based smart data source selection technique. Section IV discusses the simulation results and analysis. Finally, Section V concludes our discussion.

II. System Model

In this section, we introduce the system model, which comprises the system overview, network model, and utility model. The system overview shown in Section II-A presents a general summary of our vehicular NDN. Sections II-B and II-C present the network and utility models, which characterize our work's modeling preliminaries and objectives. Table I lists the

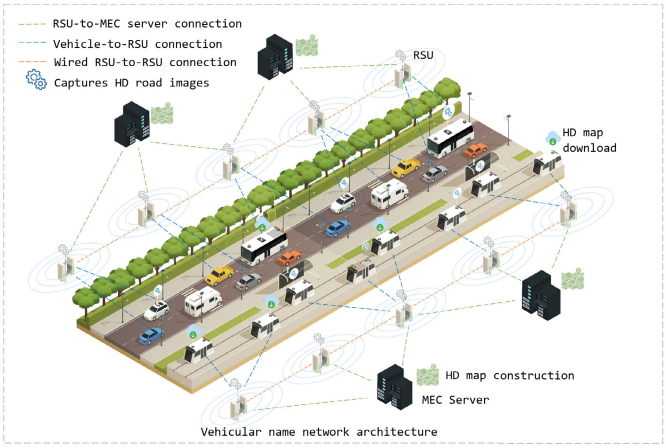


Fig. 1. Vehicular network architecture for our system model.

primary mathematical notations used in the system model in this paper.

TABLE I
LIST OF MAJOR SYMBOLS AND THEIR DEFINITIONS

Symbols	Definition
$\mathcal{V}$	Set of vehicles
$\mathcal{U}$	Set of RSUs
$\mathcal{M}$	Set of MEC servers
$v, v \in \mathcal{V}$	vehicle v
$u, u \in \mathcal{U}$	RSU u
$m, m \in \mathcal{M}$	MEC server m
N	Target cells
K	Sub-division of target cell N into
$n, n \in \mathcal{N}$	target cell n
$k, k \in \mathcal{K}$	k-th sub-division of target cell K
S	Sensing range
F	Computing power
Q	Raw environmental data
L	Coordinate location
d	Distance
a_v	Velocity of vehicle v
Π	Transmission data
R	Wireless transmission rate
t_{RTT}	Round-trip time
$H(Q_{..})$	Information entropy from data $Q_{..}$

A. System Overview

We consider a hierarchical architecture comprising vehicles, roadside units (RSUs), and MEC servers for our HD map update model [23]. In our CARLA environment, we simulate multiple vehicles equipped with sensors, communication, computing, and caching resources. Next, we set RSUs along the road equipped with sensors such as high-definition cameras to record environmental data. To ensure adequate computing resources for HD map construction, each RSU has a MEC server co-located with it. The vehicles take images of their surroundings at the start of each update period for the HD map construction process. Furthermore, the RSUs in each target cell collect vehicle information such as location, speed, sensing range, wireless transmission capability, computing capability, and optimized image data size. Using our DRL-based data source selection algorithm, the MEC server selects a vehicle from the RSU's list to engage in data transfer for HD map construction in each target cell. Next, the RSU broadcasts the MEC server's decision to the selected vehicle for image data transmission, and all surrounding vehicles enter an update mode. Vehicles with limited computational capability for the map data optimization stage rely heavily on the computation of the surrounding vehicles and only have an update mode toggled on at any target cell. Fig. 2 shows the vehicular network architecture with our proposed DSORL scheme. Additionally, we rely on the RSU to perform the image data collection if no high computational capacity vehicles are available. Also, vehicles leaving the target cell should transmit intermediate map data to the current RSUs via wireless links, which forward the map data to other RSUs via wired fronthaul connections for easy access.

B. Network Model

We consider a population of V vehicles, U RSUs, and M MEC servers, such that $\mathcal{V} = \{1, 2, \ldots, V\}, \mathcal{U} = \{1, 2, \ldots, U\}$, $\mathcal{M} = \{1, 2, \ldots, M\}$ denote a set of vehicles, RSUs, and MEC servers, respectively. Let N represent the total number of target cells C for our environment, such that $\mathcal{N} = \{1, 2, \ldots, N\}$. We characterize any given target cell $n, n \in \mathcal{N}$ with a sensing range S_n that is smaller than the overall cell range $\bar{S}$. We assume that the traffic density for n is σ_n (in vehicles/meter) [24] and the quantity of raw environmental data (in bits) is Q, as such, there will be $\sigma_n \bar{S}$ vehicles in n with Q_n environmental data at given time t. Let $v_n, v \in \mathcal{V}$ and $u_n, u \in \mathcal{U}$ represent vehicle v and RSU u belonging to target cell n, such that v_n comprises a constant velocity a_v during one update period T, a sensing range S_v, and a computing power F_v (CPU cycles/bits). We distinguish u_n by its sensing range S_u and computing power F_u. The MEC server $m \in \mathcal{M}$ has computing power F_m for the HD map construction process. Furthermore, we compute the driving direction of v_n with respect to the x-axis and with n starting from zero. We use L_{v_n} to represent the initial x-axis coordinate of v_n and $L^u_{v_n}$ is the x-coordinate of RSU u to which v_n belongs.

In our environment, we examine each vehicle's perceivable region to determine its location data. For example, v_n has a perceivable area of $[\max\{0, L_{v_n} - S_n\}, S_n + L_{v_n}]$ in n. For one update period, the real localization information of each vehicle in n with cached data is $[0, S_n + L_{v_n}]$, and the data of all other vehicles in C is $[0, \bar{S}]$. Suppose that the target cell can be subdivided into K sub-regions of equal length $\bar{S}/K$ and containing the same quantity of environmental data as Q/K. We propose a set of binary sensing variables $\hat{S}$ to indicate whether vehicles or RSUs can detect kth sub-regions in real time or store the kth environmental data. $\hat{S} = 1$ suppose v_n can detect kth region in real time or has kth stored the environmental data, otherwise $\hat{S} = 0$. As a result, we can express the sensing variable $\hat{S}_k$ for the kth sub-region as

$$\hat{S}_k = \begin{cases} 1, & S_n + L_{v_n} \geq k \cdot \bar{S}/K, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Following that, we provide a set of binary optimization variables $\mathbf{X} = \{x_{v,k}, x_{u,k}\}$, where $x_{v,k}$ and $x_{v,k}$ determine the vehicles and RSUs nearest to sub-region k for data collection. When the v_n is within k, $x_{v,k} = 1$, otherwise $x_{v,k} = 0$. Also,

we define the term $\bar{v}_n = \sum_{v=0}^{V}(x_{v,k} \cdot \hat{S}_{v,k})$, which comprises the constraint $\bar{v}_n \geq 1$, indicating that the kth sub-region contains at least one vehicle capable of data collection and $\hat{S}_{v,k}$ is the sensing variable for the kth sub-region to which vehicle v belongs. Likewise, when there are no vehicles capable of data collection ($\bar{v}_n = 0$) in k, the RSU collect and process the environmental data $Q_{u,k}$. The environmental data Q_k gathered in k is expressed as

$$Q_k = \begin{cases} Q_{v,k} = x_{v,k} \cdot \hat{S}_{v,k} \cdot \dfrac{Q}{K}, & \text{if } \bar{v}_n \geq 1, \\ Q_{u,k} = x_{u,k} \cdot \hat{S}_{u,k} \cdot \dfrac{Q}{K}, & \text{if } \bar{v}_n = 0, \end{cases} \tag{2}$$

where $Q_{v,k}$ and $Q_{u,k}$ denote the data from on v_n and u_n, respectively. $\hat{S}_{v,k}$ and $\hat{S}_{u,k}$ represent the sensing variable for vehicle v and RSU u. The time taken by vehicles and RSUs to process map data (e.g., run object detection algorithm on the image data [25]) is defined as

$$t_k = \begin{cases} t_{v,k} = \dfrac{Q_{v,k} \cdot F_{Q_{v,k}}}{F_v}, & \text{if } \bar{v}_n \geq 1, \\ t_{u,k} = \dfrac{Q_{u,k} \cdot F_{Q_{u,k}}}{F_u}, & \text{if } \bar{v}_n = 0, \end{cases} \tag{3}$$

where $F_{Q_{v,k}}$ and $F_{Q_{u,k}}$ denote the required computing intensity of $Q_{v,k}$ and $Q_{u,k}$, respectively.

Due to vehicle mobility, the new coordinate $\hat{L}_{v_n}$ of v_n after data collection is provided as

$$\hat{L}_{v_n} = L_{v_n} + a_v \cdot t_{Q_{v,k}}, \tag{4}$$

where $t_{Q_{v,k}}$ represents the time taken to collect image data by v_n in k. If v_n leaves the target cell, we send its intermediate results to the closest RSU. Consequently, the corresponding transmission data $\Pi_{Q_{v,k}}$ doubles for every increase in the number of relay hops. Also, the data $\Pi_{Q_{v,k}}$ stays unmodified at the RSU and is transmitted to the MEC server for HD map construction. However, if the RSU collects the data, we denote its transmission data as $\Pi_{Q_{u,k}}$. The corresponding transmission data Π_{Q_k} can be expressed as

$$\Pi_{Q_k} = \begin{cases} \Pi_{Q_{v,k}} = Q_{v,k} \cdot \left(|\dfrac{\hat{L}_{v_n}}{\hat{S}}| + 1 \right), & \text{if } \bar{v}_n \geq 1, \\ \Pi_{Q_{u,k}} = Q_{u,k} \cdot \left(|\dfrac{\hat{L}_{u_n}}{\hat{S}}| + 1 \right), & \text{if } \bar{v}_n = 0, \end{cases} \tag{5}$$

where $|\frac{\hat{L}_{v_n}}{\hat{S}}|$ represents the number of relay hops.

For the purpose of simplicity, the relocated distance d_{v_n} of v_n is ignored throughout the wireless transmission of intermediate results. However, d_{v_n} can be expressed as

$$d_{v_n} = \sqrt{|\hat{L}_{v_n}^{u_n} - \hat{L}_{v_n}|^2 + (D_{v,u} + H_{u_n})^2}, \tag{6}$$

where $\hat{L}_{v_n}^{u_n} = |\hat{L}_{v_n}/\hat{S}| \cdot \hat{S} + \hat{S}/2$ denotes the current coordinate of v_n in RSU u_n, $D_{v,u}$ represents the horizontal distance between v_n and RSU u_n, and H_{u_n} is the height of RSU u_n.

Moreover, we express the wireless transmission rate of v_n to RSU u_n as

$$R_{v_n}^{u_n} = b_{v_n} \cdot \log\left(1 + \frac{P_{v_n} \cdot |h_{v_n}|^2 \cdot (d_{v_n})^{-\varrho_{v_n}}}{N_{v_n}}\right), \tag{7}$$

where b_{v_n}, P_{v_n}, h_{v_n}, ϱ_{v_n}, and N_{v_n} represent v_n's allocated bandwidth, transmission power, complex channel fading coefficient, path-loss exponent, and noise power, respectively. Subsequently, we can compute the total transmission time t_{v_n} of v_n as

$$t_{v_n}^{u_n} = Q_{v,k} \cdot \left(\frac{1}{R_{v_n}^{u_n}} + \frac{|\hat{L}_{v_n}/\hat{S}|}{R_{u_n}}\right), \tag{8}$$

where R_{u_n} is the wired fronthaul link between RSUs for intermediate results sharing. Likewise, the wireless transmission rate of the RSU u_n to MEC server m can be expressed as

$$R_{u_n}^{m} = b_{u_n} \cdot \log\left(1 + \frac{P_{u_n} \cdot |h_{u_n}|^2 \cdot (d_{u_n})^{-\varrho_{u_n}}}{N_{u_n}}\right), \tag{9}$$

where b_{u_n}, P_{u_n}, h_{u_n}, ϱ_{u_n}, and N_{u_n} represent u_n's allocated bandwidth, transmission power, complex channel fading coefficient, path-loss exponent, and noise power, respectively. The total transmission time t_{u_n} of RSU u_n as

$$t_{u_n}^{m} = Q_{u,k} \cdot \left(\frac{1}{R_{v_n}^{u}} + \frac{|\hat{L}_{v_n}/\hat{S}|}{R_{u_n}}\right). \tag{10}$$

Additionally, in one update period T, the round trip time (RTT) t_{RTT} taken by either v_n or u_n can be expressed as

$$t_{RTT} = \begin{cases} t_{Q_{v,k}} + t_{v,k} + t_{v_n}^{u_n} + t_{u_n}^{m}, & \text{if } \bar{v}_n \geq 1, \\ t_{Q_{u,k}} + t_{u,k} + t_{u_n}^{m}, & \text{if } \bar{v}_n = 0, \end{cases} \tag{11}$$

where $t_{Q_{v,k}}$ and $t_{Q_{u,k}}$ denote the time taken to collect data on v_n and u_n, respectively.

Finally, to determine which car to use for any data transmission, we introduce our vehicle selection function W that depends on the round trip time t_{RTT}, vehicle's distance d_{v_n} from the RSU, velocity a_v, information entropy $H(Q_{v,k})$, and transmission data $\Pi_{Q_{v,k}}$.[1] We express W as

$$W\left(t_{RTT}, d_{v_n}, a_v, H(Q_{v,k}), \Pi_{Q_{v,k}}\right) = \lambda_1 t_{RTT} + \lambda_2 d_{v_n} + \lambda_3 a_v + \lambda_4 H(Q_{v,k}) + \lambda_5 \Pi_{Q_{v,k}}, \tag{12}$$

where $H(Q_{v,k}) = \mathbb{E}[-\log P(Q_{v,k})]$ is the information entropy (e.g., amount of detected objects) of any transmission data, λ_1, λ_2, λ_3, λ_4, and λ_5 are scoring values in $[0, 1]$ to determine the importance of each parameter. For example, we set a higher value to d_{v_n} for vehicle v_n closer to RSU u_n, and vice versa. However, it is challenging to determine these scoring values in real-time. As a result, we introduce our smart data source selection process in Section III.

C. Utility Model

Due to bandwidth limitations, it is necessary to explore methods for reducing transmission data volume and the number of transmissions between vehicles and RSUs. Therefore, the goal of this research is to minimize the total amount of transmission data and data transmissions under the HD Map

[1] We provide detailed description of each parameters in Section III-A.

update period T constraints, which can be provided by

$$\min_{x_{v,k},x_{u,k},Q_k} \quad W\left(t_{RTT}, d_{v_n}, a_v, H(Q_{v,k}), \Pi_{Q_{v,k}}\right) \tag{13a}$$

$$\text{s.t.} \quad t_{RTT} \leq T, \tag{13b}$$

$$\sum_{v=0}^{V}(x_{v,k} \cdot \hat{S}_{v,k}) \geq 1, \tag{13c}$$

$$F_{Q_{v,k}} \leq F_v, \quad F_{Q_{u,k}} \leq F_u, \tag{13d}$$

$$H(Q_{v,k}) \geq 1, \tag{13e}$$

$$d_{v_n} \leq d_{u_n}, \tag{13f}$$

$$\Pi_{Q_{v,k}} \leq R_{v_n}^{u_n}, \quad \Pi_{Q_{u,k}} \leq R_{u_n}^{m}, \tag{13g}$$

where the objective function seeks to minimize the total amount of data transmissions concerning map data size and data source selection subjected to the constraints[2] of the HD map update period T. Constraint (13b) stipulates that the entire HD map period T shall not be exceeded by the round-trip time t_{RTT}. Next, a sub-region k will always have at least one entity that is able to collecting and processing HD map data according to constraint (13c). The third constraint in (13d) ensures that the amount of resources needed for data collection and processing does not exceed the capabilities of the vehicle or RSU. Constraint (13e) requires at least one detected object in the image data. Finally, constraints (13f) and (13g) require that the separation distance between the vehicle and the RSUs is not greater than that of the RSUs and that the data transmission rate from the vehicle and RSUs not be greater than the assigned transmission rate, respectively.

III. Deep Reinforcement Learning-Based Data Source Selection Algorithm

The HD map data source selection scheme's main objective is to minimize the latency of HD map updates over the entire time-slotted system, which may be expressed as (13). The original optimization problem is extremely complicated due to the various entities involved (e.g., RSUs, vehicles, and MEC servers) and the volume of data in the long-term optimization objective [26]. As a result, using typical optimization approaches to tackle the problem directly is challenging. Therefore, we formulate the HD map data source selection as a Markov decision process (MDP) and provide a solution using appropriate reinforcement learning methods.

A. Markov Decision Process Formulation

We model the HD map data source selection issue as a MDP with states, actions, and rewards. MDPs simulate decision-making in discrete, stochastic, and sequential environments. The model focuses on an agent (e.g., a decision maker) living in an environment that changes state at random in response to the agent's action choices. The agent's immediate reward is affected by the state of the environment, as are the probabilities of future state changes. The agent's goal is to choose activities that maximize a long-term measure of total reward. Our MDP formulation can be perceived as a stochastic process comprising $\{\mathbf{s}_t, \mathbf{a}_t, p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t), \mathbf{r}_t, \mathbf{s}_{t+1}\}$, where at a time t, $\mathbf{s}_t, \forall\, \mathbf{s}_t \in \mathbf{S}_t$ represents the state space, $\mathbf{a}_t, \forall\, \mathbf{a}_t \in \mathbf{A}_t$ is the action space, $\mathbf{r}_t$ is the reward, $\mathbf{s}_{t+1}$ is the next state, and $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ represents the transition probability, respectively [27]. The return $\mathbf{R}_t$ is defined as the total of discount rewards from the present state to the end state, which can be expressed as $\mathbf{R}_t = \sum_t \gamma_t \mathbf{r}(\mathbf{s}_t, \mathbf{a}_t)$, where $\gamma \in (0, 1)$ denotes the discount factor. Based on the following definitions, we proceed to model the HD map data source selection problem using RL.

The MDP state constantly changes at any given time interval due to the various entities (e.g., vehicles, RSUs, MEC servers), entities' characteristics (e.g., mobility), and data transmissions. Therefore, we employ an RL approach to solve the MDP problem with high precision to capture such high dynamics. To decrease the complexity of our work, we examine only vehicle data selection and assume the RSU and MEC server locations stay constant. The RL approach consists of the following components.

State ($\mathbf{s}_t$)*:* At each decision time t, the environment's state $\mathbf{s}_t$ comprises the set $\mathbf{s}_t = \{\mathbf{t}, \Pi_Q, A(t), D(t), \mathbf{H}\}$ defined in a particular target cell. The various state entities are explained as follows.

- $\mathbf{t} = \{t_{RTT}^1, \ldots, t_{RTT}^v, t_{RTT}^V\}$ represents the total time required by vehicles to collect and transmit data to the MEC server. The lower the t_{RTT}, the better the network's performance, which improves system latency. The agent uses this parameter to find a suitable candidate for data transmission.
- $\Pi_Q = \{\Pi_{Q_{1,k}}, \ldots, \Pi_{Q_{v,k}}, \Pi_{Q_{V,k}}\}$ is the transmission data from the vehicles. The agent observes the data sizes and decides the most cost-efficient vehicle suitable for data transmission.
- $A(t) = \{a_{1_n}, \ldots, a_v, a_V\}$ when the data source is a vehicle, $A(t)$ is the set of driving speeds of the data sources at time t. For a vehicle v_n, $a_v > 0$ shows the same traveling direction between the requested vehicle and the RSU. Suppose the data source is an RSU, $a_v = 0$. At time t, $a_v < 0$ denotes the opposite traveling direction between the requested vehicle and the RSU. A lower speed of v_n indicates that the data source is more stable than when it is faster.
- $D(t) = \{d_{1_n}, \ldots, d_{v_n}, d_{V_n}\}$ comprises the distances of vehicles from the RSU, which can be computed from (6). For any v_n, $d_{v_n} > 0$ indicates that the vehicle is in front of the RSU, and vice versa. The agent compares the distances against other vehicles to select a suitable vehicle for the data transmission process, and the shorter the distance, the more reliable the data source.
- $\mathbf{H} = \{H(Q_{1,k}), H(Q_{2,k}), \ldots, H(Q_{V,k})\}$ is the information entropy contained in any vehicle data, which helps the agent decide the significance of various vehicle data. The more items recognized in an image, the more information it has, making it extremely helpful.

Action ($\mathbf{a}_t$)*:* We define our action as learning the corresponding importance parameter λ of the state values in a particular target cell, termed action parameters. Although there are just

[2]It is important to note that we need to normalize these constraints just in case the value difference is too large, as shown in [22].

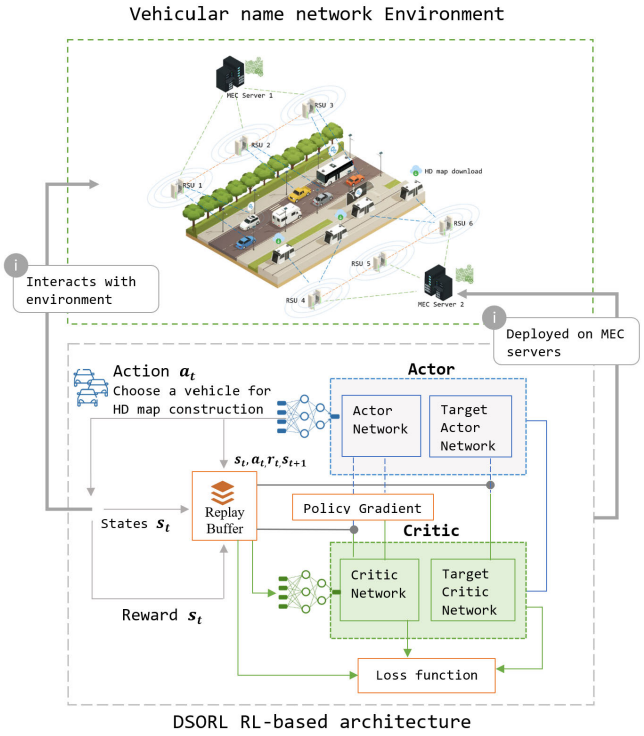


Fig. 2. DSORL framework with RL-based system and the Markov decision process.

two types of data sources (RSUs and vehicles), vehicle mobility may produce dynamic changes when employed as a data source, which signifies that the action space varies depending on the vehicle scenario. However, evaluating vehicle coverage and RSU in the connection duration time limits the amount of data sources. For simplicity, we do not consider when the vehicle leaves the particular target cell. However, we provide each vehicle the option to determine its action space by using previous action values stored on the RSU in a target cell or by using its inherent actions. We begin by calculating the optimal values for each state $\mathbf{s}_t^* = \left\{\mathbf{t}^{min}, \Pi_Q^{min}, A^{min}, D^{min}, \mathbf{H}^{max}\right\}$ based on the preceding state definition. The rank of the chosen data source with action $\mathbf{a}_t$ may then be determined as follows:

$$G_{v_n,t} = \lambda_1 \frac{t_{RTT}^v}{t_{RTT}^{min}} + \lambda_2 \frac{d_{v_n}}{d_{v_n}^{min}} + \lambda_3 \frac{a_v}{a_v^{min}} + \lambda_4 \frac{H(Q_{v,k})}{H^{max}(Q_{v,k})} + \lambda_5 \frac{\Pi_{Q_{v,k}}}{\Pi_{Q_{v,k}}^{min}}, \tag{14}$$

where $\text{Max}\{G_{v_n,t}\}$ denotes that the highest rank of data source v_n is chosen when the learned action parameters $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$ are applied to the state $\mathbf{s}_t$ values. For example, consider the round-trip time t_{RTT} of vehicles in (14), our RL agent seeks to select the vehicle with the shortest t_{RTT}. As a result, we designed its action to penalize any vehicle with a high t_{RTT} by choosing a smaller action parameter while boosting others with a higher action parameter to enhance the chances of getting selected. Furthermore, we mention that the motivation for considering the highest rank of the data source using $\text{Max}\{G_{v_n,t}\}$ is to encourage the selection of vehicles closer to the optimal value $\mathbf{s}_t^*$. Therefore, by punishing other vehicles with less optimal values using our action parameters, we can ensure that the highest-ranked data source will be the optimal or near-optimal data source. Also, by altering the action parameters to impact the choice of data sources, we intend to allow RL-based learning of the consequences of a broader range of states. The value ranges of action parameters in this work are $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{0, 0.2, 0.6, 0.8, 1\}$ and $\{0, 0.25, 0.5, 0.75\}$. Finally, when the agent takes action $\mathbf{a}_t$ under state $\mathbf{s}_t$, the chosen data source is the highest ranked $G_{v_n,t}$.

Reward ***($\mathbf{r}_t$):*** We return the corresponding reward once each action is completed to guarantee that the RL model can learn from previous experience, which characterizes the overall benefit of an agent adhering to a policy. To comprehend our agent's reward, consider the following design principle: 1) increase throughput to the greatest extent practicable. Throughput is the most essential metric of map data transmission, which signifies that the vehicle can send and acquire map data rapidly and effectively, 2) improve transmission time via de-congestion. The goal is to prevent the added expense of frequent vehicle requests and data source handovers, which keeps the connection steady and increases throughput, and 3) reduce transmission delays through efficient data source selection. HD map updates have higher requirements for transmission latency in the autonomous driving scenario. Due to the reduced latency, the data source may provide map data quickly, decreasing the data package wait time. At time t, the agent observes state $\mathbf{s}_t$ and then takes action $\mathbf{a}_t$, following policy π to obtain a reward $\mathbf{r}_t$, which can be expressed as

$$\mathbf{r}_t = \delta_1 \mathcal{T}(R_{v_n}^{u_n}) - \delta_2 \mathcal{C}(\Pi_Q) - \delta_3 \Omega(t_{RTT}), \tag{15}$$

where $\mathcal{T}(R_{v_n}^{u_n})$ is the average transmission throughput, which is the quantity of data successfully transferred from vehicles to the RSU in a given period and is commonly measured in bits per second (bps). The average number of selected transmission data is denoted by $\mathcal{C}(\Pi_Q)$, and the smaller the $\mathcal{C}(\Pi_Q)$, the higher the reward. The average RTT time for all packets during transmission is $\Omega(t_{RTT})$. We employ the impacting factors δ_1, δ_2, and δ_3 to weigh these metrics, which gives a reasonable reward $\mathbf{r}_t$, and the ranges of these parameters are $1 \leq \delta_1 \leq 2$, $0 \leq \delta_2 \leq 2$, and $0 \leq \delta_3 \leq 0.5$.

B. Value Function and Policy

The RL algorithm evaluates an agent's performance in a given state using state-value functions (or action value functions). We adopt the Bellman expectation equation to characterize the value function as a discounted expected return [27], which can be expressed as

We model an MDP to find an optimal policy π^* that minimizes the cumulative HD map update's latency in a given time-slot T. The agent's action $\mathbf{a}$ accompanies probability distribution $\mathcal{P}$ and parameter θ at state s to evaluate stochastic policy function π_θ at a given time step t, which can be denoted as

$$\mathbf{V}(\mathbf{s}_t) = \mathbf{r}_t(\mathbf{s}_t, \mathbf{a}_t) + \gamma \sum_{\mathbf{s}_t} \mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\mathbf{V}(\mathbf{s}_{t+1}). \tag{16}$$

Based on the Bellman's optimal equation, we can compute the the total maximum discounted reward $\mathbf{V}^*(\mathbf{s}_t)$ iteratively as $\mathbf{V}^*(\mathbf{s}_t) = \max_{\mathbf{a}_t} \mathbf{V}(\mathbf{s}_t)$ [27] and the state-value function's convergence yield the optimal policy π^* calculated as $\pi^* = \arg\max_{\mathbf{a}_t} \mathbf{V}(\mathbf{s}_t)$.

Usually, we require prior environment information when employing a model-based reinforcement learning (RL) algorithm. In this work, the reward and transition probabilities are unknown to our environment. As a result, we implement a model-free RL algorithm. Q-learning algorithm is a well-known model-free RL [27]. In a discrete state space MDP, the agent iteratively learns Q-values stored in the lookup table. The Q-value $\mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t)$ update is shown as [28]

$$\mathcal{Q}_t = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_a \mathcal{Q}(\mathbf{s}_{t+1}, a), \tag{17}$$

$$\mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow \mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t) + \alpha(\mathcal{Q}_t - \mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t)). \tag{18}$$

where $\alpha \in [0, 1]$ is the learning rate. The error between target value Q_t and predicted value $Q(\mathbf{s}_t, \mathbf{a}_t)$ is expressed as a time-difference error (TD-error). It is important to note that we can obtain the optimal Q-values policy upon convergence

C. Deep Reinforcement Learning Based Smart Data Source Selection Process

Model-free reinforcement learning algorithms can be categorized into value-based and policy-based techniques based on policy updates. By learning a value function, value-based approaches enable agents to choose the best policy (e.g., Q learning). Also, because the agent has a wide variety of action parameters to select from, the action space in our work is continuous. A naive method is to discretize the action space using a value-based procedure in the continuous domain, which results in the curse of dimensionality and the loss of critical information about the structure of the action domain. The policy-based approaches use parameterized policies to learn stochastic policies for high-dimensional and continuous action space problems.

1) Policy Based Method: At state $\mathbf{s}_t$, action $\mathbf{a}_t$ follows the probability distribution with parameter θ, and we can express the stochastic policy function π_θ at time step t for the parameterized policy as

$$\pi(\mathbf{a}_t|\mathbf{s}_t, \theta) = \mathcal{P}\left\{\mathbf{A}_t = \mathbf{a}|\mathbf{S}_t = \mathbf{s}, \theta^t = \theta\right\}. \tag{19}$$

The objective function $\mathbb{J}(\pi)$ is expressed as

$$\mathbb{J}(\pi) = \mathbb{E}_{\mathbf{s}\sim\rho_\pi, \mathbf{a}\sim\pi_\theta}[\sum_t \mathbf{r}(\mathbf{s}_t, \mathbf{a}_t)], \tag{20}$$

which denotes the expected return, and ρ_θ represents the policy π's discounted state distribution's probability. The Policy gradient approach [29] determines the optimal parameter π^* using the steepest descent and performs parameter updates as follows:

$$\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} \mathbb{J}(\pi_t). \tag{21}$$

When the action is a high-dimensional vector, the stochastic policy gradient (SPG) requires a significant amount of computing to execute action sampling for the stochastic policy. The deterministic policy gradient (DPG) [30], on the other hand, explicitly provides deterministic behavior policies while avoiding frequent action sampling. The DPG objective function's gradient is specified as

$$\nabla_\theta \mathbb{J}(\vartheta_\theta) = \mathbb{E}_{\mathbf{s}\sim\rho^\vartheta}[\nabla_\theta \vartheta_\theta(\mathbf{s}) \nabla_{\mathbf{a}} \mathcal{Q}_\vartheta(\mathbf{s}, \mathbf{a})|_{\mathbf{a}=\vartheta_\theta(\mathbf{s})}]. \tag{22}$$

DPG-based approaches, on the other hand, generate deterministic strategies without investigating the environment, which results in an off-policy balance in state-action exploitation and exploration. The behavior policy adopts a stochastic policy to ensure sufficient action exploration. Conversely, the target policy is deterministic, which capitalizes on the full benefit of an efficient deterministic policy. Hence, the DPG method's learning structure follows the actor-critic (AC) approach, as explained in the subsequent paragraph.

2) Actor-Critic Approach: The actor-critic approach integrates the advantages of policy-based and value-based approaches. In particular, the actor creates action given a state via a policy function. The critic generates the action value function and utilizes TD-error (loss function) to analyze the action's performance. The actor then employs the DPG technique to update the policy parameter with the critic's output. Next, the critic applies the gradient descent approach to update the action value function [31].

The function approximators given as $\theta^{\mathcal{Q}}$ and θ^ϑ are applied as the action-value and policy function. The value function update is expressed as

$$\varphi_t = \mathbf{r}_t + \gamma \mathcal{Q}(\mathbf{s}_{t+1}, \vartheta(\mathbf{s}_{t+1}|\theta^\vartheta)) - \mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t|\theta^{\mathcal{Q}}), \tag{23}$$

where the future $\theta^{\mathcal{Q}}$ can be calculated using the expression

$$\theta^{\mathcal{Q}}_{(t+1)} = \theta^{\mathcal{Q}}(t) + \alpha_c \varphi_t \nabla_{\theta^{\mathcal{Q}}} \mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t|\theta^{\mathcal{Q}}). \tag{24}$$

Using the DPG approach, the actor updates the policy parameters θ^ϑ:

$$\theta^\vartheta_{(t+1)} = \theta^\vartheta(t) + \alpha_a \nabla_{\theta^\vartheta} \vartheta(\mathbf{s}_t|\theta^\vartheta) \nabla_a \mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t|\theta^{\mathcal{Q}})|_{a=\vartheta(\mathbf{s}_t)}. \tag{25}$$

3) DSORL: A Deep Deterministic Policy Gradient-Based Data Source Selection Algorithm: Deep deterministic policy gradient (DDPG) is a DRL technique that combines the advantages of Q-learning and policy gradient approaches [32]. DDPG is a good choice for our environment with a continuous action space. It can efficiently solve sequence decision-making problems due to its ability to directly output actions, convergence stability, lower sensitivity to hyperparameters, and reduced computational complexity. A typical DDPG comprises two models: actor and critic, which form its actor-critic technique. The actor consists of a policy network that uses states as inputs to produce discrete or continuous actions instead of a probability distribution over actions. The critic forms a Q-value network that utilizes the state and action as input to produce the Q-value for criticizing the performance of an action with the help of TD-error (loss function). The actor updates the policy parameter with the critic's output via a deterministic policy gradient (DPG) method [31]. DPG immediately creates a deterministic behavior policy and skips numerous action sampling. The critic applies a gradient descent method to update the action-value functions.

Algorithm 1 DSORL: DDPG-based algorithm for smart data source selection process

input: actor $\vartheta(\mathbf{s}_t|\theta^{\vartheta})$, critic $\mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t|\theta^{\mathcal{Q}})$, learning rates: α_a, α_c, discount parameter γ, smooth update τ, and buffer D;

Initialize θ^{ϑ}, $\theta^{\mathcal{Q}}$, $\theta^{\vartheta'} \leftarrow \theta^{\vartheta}$, and $\theta^{\mathcal{Q}'} \leftarrow \theta^{\mathcal{Q}}$;
void `execute_DSORL(states)`**:**
  for epoch $n = 0$; $n \leq N$; $n++$ **do**
    Initialize a random process $\mathcal{M}$;
    Initialize initial state s_0;
    for period $t = 0$; $t \leq T$; $t++$ **do**
      Select an action $\mathbf{a}_t = \vartheta(\mathbf{s}_t|\theta^{\vartheta}) + \mathcal{M}_t$;
      Execute $\mathbf{a}_t$, $\mathbf{s}_{t+1}$, and $\mathbf{r}_t$;
      Store $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{s}_{t+1}) \rightarrow D$;
      Sample a random d from D;
      Initialize TD-error: $\vartheta\mathbf{a}_t = r_t + \gamma\mathcal{Q}(\mathbf{s}_{t+1}, \vartheta(\mathbf{s}_{t+1}|\theta^{\vartheta})) - \mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t|\theta^{\mathcal{Q}})$;
      Update critic: $\theta^{\mathcal{Q}}_{t+1} = \theta^{\mathcal{Q}}_t + \alpha_c \frac{1}{d}\vartheta\mathbf{a}_t \nabla_{\theta^{\mathcal{Q}}}\mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t|\theta^{\mathcal{Q}})$;
      Compute policy gradient: $\mathcal{J} = \frac{1}{d}\sum_t^T \nabla_a \mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t|\theta^{\mathcal{Q}})|_{a=\vartheta(\mathbf{s}_t)} \nabla_{\theta^{\vartheta}}\vartheta(\mathbf{s}_t|\theta^{\vartheta})$;
      Update actor: $\theta^{\vartheta}_{t+1} = \theta^{\vartheta}\mathbf{a}_t + \alpha_a \nabla_{\theta^{\vartheta}}\mathcal{J}$;
      Update target network:
        $\theta^{\vartheta'} \leftarrow \tau\theta^{\vartheta} + (1-\tau)\theta^{\vartheta'}$
        $\theta^{\mathcal{Q}'} \leftarrow \tau\theta^{\mathcal{Q}} + (1-\tau)\theta^{\mathcal{Q}'}$;
    end
  end
return
Function `main()`**:**
  Initialize `CARLA_env`;
  Generate traffic flows
  `/* transmit environmental data */`
  data_sources`[]` = run `carla_ros_bridge()`;
  `execute_DSORL(data_sources)`;
return

Usually, applying function approximators directly to the actor-critic method coupled with the deep neural network is unstable due to consecutive shared parameters [33]. However, training a DQN network with experience replay breaks this shared similarity. Therefore, the authors in [34] introduced a DDPG algorithm that combines the advantages of the actor-critic approaches and DQN with experience replay, which efficiently runs over the continuous action spaces.

Experience Replay: As a result of the agent and environment interactions, the data tuples $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{s}_{t+1})$ is produced and stored in replay a buffer B. In addition, the critic and actor randomly utilize a minibatch b sample $(\mathbf{s}_b, \mathbf{a}_b, \mathbf{r}_b, \mathbf{s}_{b+1})$ from the buffer for the value function and policy function parameter updates.

Target Network: In the work [35], the authors proved that Q-learning is unstable when directly implemented with deep neural networks due to parameter sharing between the target network and the predicted network. As a result, we use

Fig. 3. CARLA vehicular NDN environment setup with DSORL process.

replicates of actor $\vartheta'(\mathbf{s}_t|\theta^{\vartheta'})$ and critic network $\mathcal{Q}'(s, a|\theta^{\mathcal{Q}'})$ to evaluate the target value. Additionally, soft updates to target network weights are applied to improve the training stability. Finally, Algorithm 1 presents the DDPG-based algorithm to our DSORL approach.

IV. Experiment Results and Analysis

In this section, we first introduce the system configuration for our experiments in Section IV-A. Secondly, we present the performance metrics and experiment benchmarks for the basis of our experiment in Section IV-B. Finally, we conduct comprehensive experiments to evaluate the performance improvements of the proposed mechanism and validate our results in Section IV-C.

A. System Configuration

Fig. 3 shows our vehicular NDN environment with our proposed data source selection scheme. In Fig. 3, we create multiple vehicles that send data transmission requests to the MEC server, which comprises the DSORL algorithm. Also, Fig. 3 shows a sample DSORL code running on the MEC server to select vehicles suitable for data transmission.

We conduct the research using the Python 3.8 environment on a Core i7 CPU machine with a 3.9GHz clock speed and 64GB of RAM. Using the CARLA simulation environment, we deploy the proposed DSORL algorithm for HD map updates in vehicular settings. CARLA facilitates the creation, training, and validation of autonomous driving systems and provides open digital assets (urban layouts, buildings, and vehicles), open-source protocols, and technologies. The simulation platform provides dynamic sensor packages, ambient conditions, comprehensive control of static

TABLE II
EXPERIMENT PARAMETER LIST

Parameters	Value
Number of vehicles	50
Number of RSU	20
Number of routers	10
MEC servers	5
Communication range	$200 - 300$m
P2P link delay/ bandwidth	10ms/200Mbps
CARLA version	0.9.12
Environment setting	Town 1
Traffic	50 pedestrians/30 cyclists
Road speed	$30 - 60$m/s
Learning rate α_a, α_c	0.001
Impact factors $\{\delta_1, \delta_2, \delta_3\}$	1, 0.5, and 0.5
Discount factor γ	0.9
Soft update parameter τ	0.01
Size of replay buffer and mini-batch	1000 and 64
Transportation protocol	IEEE 802.11ac
Map data size	$50 - 400$MB

and dynamic actors, and more. We employ a multilane urban traffic flow vehicle trajectory, including vehicles, pedestrians, crossings, cross traffic, traffic laws, and other complexities that differentiate urban driving from track racing. In addition, we connected a robot operating system (ROS) with CARLA (ROSbridge) to simulate computations on multiple vehicles and RSUs, such as router and RSU connections, HD map data collection, optimization, and construction. Using ROS, vehicles in our CARLA environment can also execute complicated algorithms, such as the YOLOX object detection algorithm [25]. Multiple Nvidea Jetson AGX models with a high processing capability are utilized on the MEC server for HD map creation and dissemination. Also, we assume the P2P link latency between the router, vehicles, RSU, and MEC server to be 10 milliseconds with a bandwidth of 200 Mb/s. Then, using ROSbridge, we transmit the vehicle data from CARLA, including the number of vehicles, driving direction, speed, and road network. In addition, we implement our DRL method using the OpenAI Gym [35] tools. To boost the efficiency of HD map transmission, we employ the IEEE 802.11ac protocol to send the map data. Table II presents the parameter values in detail.

B. Performance Metrics and Experiment Benchmarks

In this part, we introduce the prerequisites for Section IV-C, including baseline schemes and performance metrics. The experiment baseline schemes and performance metrics describe how the many associated approaches for this topic were selected and the measurement metrics utilized to get these findings, respectively.

1) Baseline Schemes (Benchmarks): To show the performance of our proposed DSORL scheme, we implement the three equivalent baseline schemes listed below:

1) *RLSS Technique:* This approach employs a double deep Q-network (Double DQN) learning-based architecture to train a neural network as an agent to decide on data source selection to improve HD map update action performance concerning latency, throughput, and packet loss.
2) *HDM-RTT:* This technique combines an HD map and a random tree sampling-based algorithm to quickly obtain high-quality and feasible map trajectories in complex campus scenarios.
3) *Pro-RTT:* In this system, the vehicle employs the probability-based handover approach to choose a new data source by monitoring the RTT, which decreases the frequency of handovers.

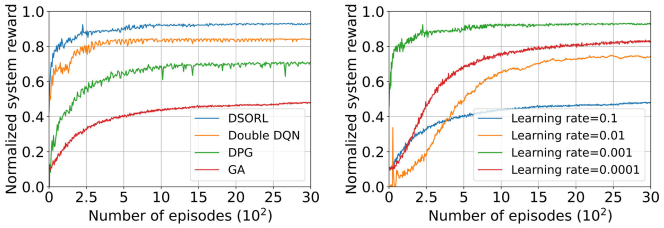
2) Metrics of Performance: To evaluate the performance of our proposed DSORL scheme, we employ the following performance metrics:

1) *Throughput:* The amount of successfully received map data divided by the transmission time is referred to as throughput. This metric applies to the overall stages involved in our latency optimization scheme.
2) *Transmission Time:* This is the time it takes from the start of a map transmission to the finish, including data collection, object detection, data transfer, and HD map update.
3) *Packet Loss Rate:* This is derived by dividing the number of lost packets by the total number of packets sent.
4) *Handover Times:* This metric displays how many times the RSU exchanges data sources throughout the HD map transmission procedure. Data transmission efficiency will be reduced if data sources are switched often.

C. Implementation Discussions

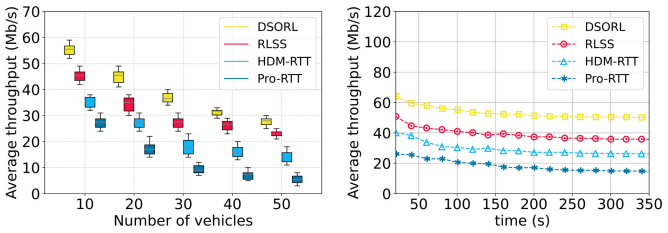
In this section, we present and explain the various results obtained in our experiment using the previously introduced baseline schemes and performance metrics.

1) Convergence Analysis: In this experiment, we evaluate the convergence performance of our proposed DSORL method for smart data source selection with a greedy approach (GA), deterministic policy gradient (DPG), and Double DQN, taking into account normalized reward and variable learning rates, as seen in Fig. 4. Fig. 4a depicts the performance of DSORL, Double DQN, DPG, and GA based on normalized reward convergence, and Fig. 4b shows the impact of varied learning rates on the convergence of the DSORL algorithm. According to Fig. 4a, all algorithms converge, with the proposed algorithm achieving the fastest convergence at around 450 epochs and the highest normalized reward at almost 0.95. The observed trend can be attributed to DSORL properties that significantly improve the learning process. For example, DSORL uses deep neural networks, allowing it to handle high-dimensional observation spaces compared to the similar architecture in DPG. Additionally, DSORL outperforms Double DQN due to its policy-based algorithm, which can handle both continuous and discrete action spaces and is less sensitive to hyperparameters. Moreover, DSORL uses a single neural network and is less prone to convergence issues, while Double DQN requires two separate networks and can have difficulty converging. Furthermore, GA chooses actions greedily, which results in the worst convergence with a normalized reward of around 0.65. GA appears to be appropriate for nonlinear integer programming (NLIP) problems that cannot capture the high dynamics of the MEC system.



(a) Reward convergence (b) Varied learning rates

Fig. 4. Convergence analysis.



(a) Throughput (Mb/s) vs vehicles (b) Throughput (Mb/s) vs time (s)

Fig. 5. Throughput analysis.

We examine the convergence of the proposed DSORL algorithm, with the fastest convergence and the largest reward value, using several learning rates such as $\alpha = 0.1$, $\alpha = 0.01$, $\alpha = 0.001$, and $\alpha = 0.0001$. The convergence for each learning rate is shown in Fig. 4b, with $\alpha = 0.001$ achieving the maximum reward. We can conclude that using $\alpha = 0.001$ as the learning rate for our suggested method results in improved convergence. However, this is not true in every circumstance because the choice of α is dependent on the algorithm and the environment.

2) Average Throughput Analysis: We first discuss the impact of the DRL algorithm on data source selection, which allows the DSORL scheme to adapt to the dynamic and complex nature of vehicular NDN. The DRL algorithm's ability to learn from the system's current state, considering factors such as latency, throughput, and packet loss, allows it to select suitable vehicles for HD map data transmission to MEC servers, thereby optimizing the performance of the vehicular NDN environment. The effectiveness of the DSORL method is demonstrated through a comparison with various baseline schemes (RLSS, Pro-RTT, and HDM-RTT) in terms of average throughput performance for a varying number of vehicles, as shown in Fig. 5a. As seen in Fig. 5a, increasing the number of vehicles decreases the average throughput of the system. However, the DSORL scheme significantly outperforms other baseline systems, owing to reduced latency, packet loss, and bandwidth utilization resulting from fewer data transmissions. For example, when the number of vehicles is $10 - 30$, the median average throughput is around 8.2, 15.8, 20.5, and 35 (Mb/s) in the corresponding HDM-RTT, Pro-RTT, RLSS, and DSORL schemes, respectively.

Similarly, Fig. 5b shows the average throughput with time. As shown in Fig. 5b, the average throughput decreases with time. DSORL achieves a higher throughput, while all baseline schemes maintain a relatively steady state. Considering 100s to 200s duration, DSORL obtains an average throughput of 56.5 (Mb/s), RLSS, HDM-RTT, and Pro-RTT, and 39, 30, and 20 (Mb/s), respectively. Our detailed experimental results demonstrate that the proposed DSORL scheme effectively optimizes data source selection for HD map data transmission in vehicular NDN environments. The method's RL-based MDP formulation and adaptive capabilities allow it to outperform existing baseline schemes, making it a promising solution for real-world vehicular network applications.

3) Packet Loss Analysis: In this section, we highlight the impact of our DSORL algorithm on minimizing packet loss in the data source selection process. The DSORL algorithm's ability to learn from the system's current state, considering factors such as network congestion and interference between vehicles, allows it to select suitable vehicles for HD map data transmission to MEC servers, thereby reducing the packet loss rate.

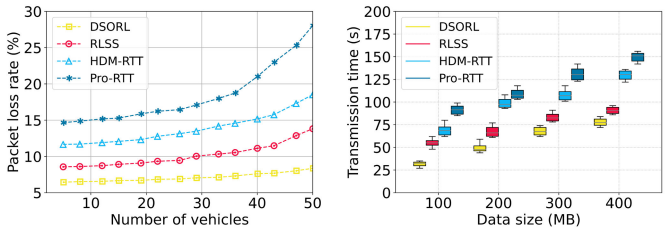
To demonstrate the effectiveness of the DSORL method in minimizing packet loss, we compare it with various baseline schemes (RLSS, Pro-RTT, and HDM-RTT) in Fig. 6a. Fig. 6b shows the average packet loss rate for each scheme, revealing two important observations:

1) the DSORL scheme significantly outperforms other baseline systems, and
2) when the number of vehicles exceeds 20, the packet loss rate of RLSS, Pro-RTT, and HDM-RTT schemes increases substantially compared to the DSORL.

For example, the average packet loss rate for DSORL, RLSS, Pro-RTT, and HDM-RTT schemes is 4.95%, 8.95%, 12.5%, and 15.65%, respectively. The superior packet loss rate of DSORL can be attributed to its ability to reduce network congestion, thereby improving data transfer quality and mitigating interference between multiple vehicles in the environment. Our proposed DSORL scheme is highly effective in optimizing data source selection for HD map data transmission in vehicular NDN environments, particularly regarding packet loss rate. The RL-based MDP formulation and adaptive capabilities allow the DSORL method to outperform existing baseline schemes, making it a promising solution for real-world vehicular network applications. The superior performance of DSORL in minimizing packet loss rate highlights its practical implications and relevance for real-life scenarios, where maintaining a low packet loss rate is crucial for ensuring reliable communication and data transfer in vehicular networks.

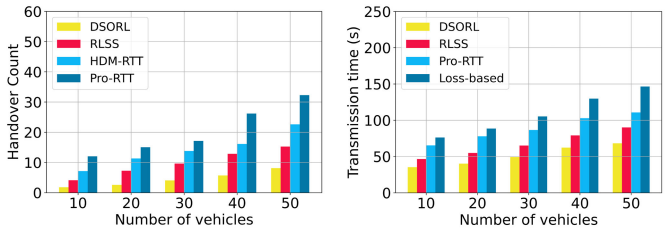
4) HD Map Data Size Analysis: In this section of the experiment, we investigate further the impact of HD map data size on transmission time by comparing the proposed DSORL method with various baseline techniques (RLSS, Pro-RTT, and HDM-RTT). We present the transmission time for each baseline system when the number of vehicles is 20, and the data size varies, as shown in Fig. 6b. From our analysis, we make two key observations:

1) the DSORL scheme consistently outperforms the other baseline systems across different data sizes; and
2) as the data size increases, the transmission time for the DSORL method grows slower than the other baseline schemes.



(a) Packet loss rate (%) (b) Transmission data size (MB/s)

Fig. 6. Packet loss rate and transmission data size.



(a) Handover count (b) Transmission time (s)

Fig. 7. Handover count and transmission time (s).

The DSORL method's superior performance can be attributed to its integrated image data optimization, which reduces the amount of data transmitted and, consequently, the transmission time. This approach is particularly crucial in NDN vehicular scenarios, where data size significantly impacts transmission time performance. For example, when the data size increases from 50MB to 200MB, the median transmission time rises by 32, 91, 132, and 145s for the DSORL, RLSS, Pro-RTT, and HDM-RTT schemes, respectively. Our analysis demonstrates the practical relevance of the proposed DSORL method in real vehicular network environments, as it can effectively reduce transmission time while maintaining high-quality data transfers. This process is particularly significant for ensuring the efficiency and reliability of HD map transmissions in autonomous driving scenarios.

5) Transmission Time and Handover Count Analysis: In the following result discussion, we examine how well the DSORL scheme performs compared to different baseline techniques, focusing on transmission time and handover count of different vehicle densities. Figs. 7 and 7b depict the handover count and transmission time of each method in these aspects, respectively. Our analysis leads to three critical observations:

1) the DSORL scheme demonstrates superior performance compared to the other baseline schemes;
2) as the number of vehicles increases, the transmission time and handover count in RLSS, Pro-RTT, and HDM-RTT schemes show a significant rise, whereas the DSORL method remains relatively stable; and
3) there is a direct correlation between handover count (number of data source switches) and transmission data size on transmission time, which further validates the effectiveness of our proposed DSORL scheme.

The primary reason behind DSORL's superior performance lies in its ability to minimize the number of handovers, thereby increasing connection stability. Our DSORL approach achieves this reduction, which effectively reduces network congestion and improves data transmissions. In real-world vehicular network environments, minimizing handovers is essential for maintaining a stable connection and enabling efficient data transmission, particularly in dense traffic scenarios.

Furthermore, to provide more context, let us consider an example with 20 vehicles in the environment. The average handover count for DSORL, RLSS, Pro-RTT, and HDM-RTT systems is 4.9, 9.2, 12.7, and 16.5, respectively. The average transmission time for DSORL, RLSS, Pro-RTT, and HDM-RTT schemes is 4.27, 8.6, 14.7, and 19.8s, respectively. These results further emphasize the practical relevance of the DSORL scheme and its potential to optimize HD map transmissions in autonomous driving scenarios.

6) Communication Cost Analysis: In this section, we analyze the impact of the number of vehicles and available bandwidth on the communication cost of our proposed DSORL scheme, as illustrated in Fig. 8a. The communication cost is a crucial factor for evaluating the effectiveness of DSORL in real-world vehicular network scenarios. From Fig. 8a, we can derive three significant observations:

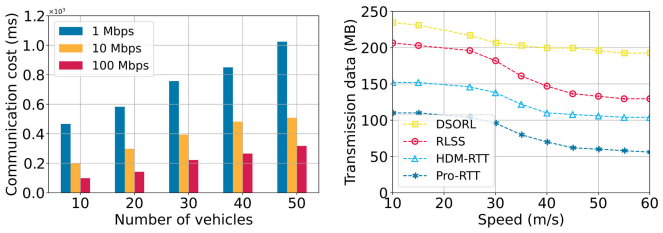
1) the communication cost of DSORL increases with the growth in the number of vehicles,
2) the communication cost escalates as the bandwidth decreases, and
3) the DSORL scheme can effectively manage the communication cost under various bandwidths, maintaining acceptable performance even at lower bandwidths.

This analysis demonstrates the practical implications of the DSORL scheme in real-world vehicular network environments.

Bandwidth plays a critical role in the map update process, as limited bandwidth may lead to increased communication costs, impacting the system's overall performance. For instance, when there are 20 vehicles, the communication cost of DSORL is 2.4, 4.7, 46.9, and 468.8ms at 100, 10, and 1Mb bandwidths, respectively, when considering different bandwidths. As a result, we employ the IEEE 802.11ac standard, which offers higher data rates, instead of the IEEE 802.11p (3 Mbit/s), to ensure that the HD map update process in DSORL does not adversely affect the system's performance. These insights help demonstrate the robustness and adaptability of DSORL in practical vehicular network environments, emphasizing its potential to manage communication costs effectively and maintain satisfactory performance under different conditions.

7) Vehicle Speed Analysis: To further investigate the effectiveness of the DSORL scheme, we analyze the impact of vehicle driving speed on transmission data size for various baseline approaches. Fig. 8b shows the relationship between transmission data size and vehicle speed, ranging from 10 to 60m/s for 30 vehicles. The following key observations can be made:

1) The DSORL scheme consistently outperforms the other baseline methods at various speeds, demonstrating its adaptability and effectiveness in dynamic vehicular network scenarios.



(a) Communication cost (b) Transmission data vs speed

Fig. 8. Communication cost and speed impact.

2) As vehicle speed increases, the transmission data size decreases for all schemes. This observation highlights the importance of considering vehicle speed as a factor influencing data transmission performance in NDN vehicular environments.
3) A substantial reduction in transmission data size is observed for all baseline methods at a speed of 30m/s. For instance, the transmission data sizes for RLSS, Pro-RTT, and HDM-RTT methods are 182, 138, and 120MB, respectively. However, the DSORL scheme maintains a consistent decrease of 207MB, showcasing its robustness in managing transmission data size under varying speeds.
4) The reduced transmission time and vehicle handovers in the DSORL scheme allow for a more efficient and stable data transmission process, regardless of the vehicle's speed. This finding emphasizes the practical relevance of our proposed method in real-world vehicular network environments.

By incorporating these additional insights, we can conclude that our discussion highlights the versatility and effectiveness of the DSORL scheme in managing the complexities of real vehicular network scenarios.

V. Conclusion

In this study, we designed and implemented a smart data source selection scheme for HD map updates in vehicular NDN scenarios. We created a vehicular NDN environment with the CARLA simulator and ROS2 to collect environmental data using AV sensors. Next, considering our vehicular NDN's dynamic and complex nature, we formulated the data source selection problem as an MDP and solved it using a DRL-based approach. For simplicity, we termed our proposed scheme DSORL, which selects suitable vehicles for HD map data transmission to MEC servers. DSORL takes advantage of the NDN architecture to effectively handle large-scale HD map delivery in vehicular scenarios and selects suitable data sources in real-time to stay current with dynamic and complicated environments. The experiment results indicated that our suggested method outperformed existing baseline schemes across all performance criteria in the evaluation. For instance, the system throughput increases by $65\% - 72.68\%$ compared to other baseline systems. Similarly, the proposed approach can minimize packet loss rate, data size, and transmission time by up to 60.6%, 77.5%, and 54.1%, respectively.

References

[1] L. Liu, Y. Zhou, V. Garcia, L. Tian, and J. Shi, "Load aware joint CoMP clustering and inter-cell resource scheduling in heterogeneous ultra dense cellular networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2741–2755, Mar. 2018.

[2] S. Bijjahalli and R. Sabatini, "A high-integrity and low-cost navigation system for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 356–369, Jan. 2021.

[3] Z. Zhang et al., "An overview of security support in named data networking," *IEEE Commun. Mag.*, vol. 56, no. 11, pp. 62–68, Nov. 2018.

[4] D. Grewe, M. Wagner, and H. Frey, "A domain-specific comparison of information-centric networking architectures for connected vehicles," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2372–2388, 3rd Quart., 2018.

[5] M. Amadeo, C. Campolo, and A. Molinaro, "Information-centric networking for connected vehicles: A survey and future perspectives," *IEEE Commun. Mag.*, vol. 54, no. 2, pp. 98–104, Feb. 2016.

[6] R. W. L. Coutinho, A. Boukerche, and A. A. F. Loureiro, "Design guidelines for information-centric connected and autonomous vehicles," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 85–91, Oct. 2018.

[7] J. Wang, C. Jiang, K. Zhang, T. Q. S. Quek, Y. Ren, and L. Hanzo, "Vehicular sensing networks in a smart city: Principles, technologies and applications," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 122–132, Feb. 2018.

[8] Y. Xu, H. Zhou, T. Ma, J. Zhao, B. Qian, and X. Shen, "Leveraging multiagent learning for automated vehicles scheduling at nonsignalized intersections," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11427–11439, Jul. 2021.

[9] W. Xu et al., "Internet of Vehicles in big data era," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 19–35, Jan. 2018.

[10] F. Sun et al., "Cooperative task scheduling for computation offloading in vehicular cloud," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11049–11061, Nov. 2018.

[11] Y. Peng, L. Liu, Y. Zhou, J. Shi, and J. Li, "Deep reinforcement learning-based dynamic service migration in vehicular networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[12] D. Chen, Y.-C. Liu, B. Kim, J. Xie, C. S. Hong, and Z. Han, "Edge computing resources reservation in vehicular networks: A meta-learning approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5634–5646, May 2020.

[13] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[14] W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, "MapTask scheduling in MapReduce with data locality: Throughput and heavy-traffic optimality," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 190–203, Feb. 2016.

[15] P. Li et al., "Traffic-aware geo-distributed big data analytics with predictable job completion time," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1785–1796, Jun. 2017.

[16] A. G. Anadiotis, G. Morabito, and S. Palazzo, "An SDN-assisted framework for optimal deployment of MapReduce functions in WSNs," *IEEE Trans. Mobile Comput.*, vol. 15, no. 9, pp. 2165–2178, Sep. 2016.

[17] K. Jo, C. Kim, and M. Sunwoo, "Simultaneous localization and map change update for the high definition map-based autonomous driving car," *Sensors*, vol. 18, no. 9, p. 3145, Sep. 2018.

[18] B. Camburn et al., "Computer-aided mind map generation via crowdsourcing and machine learning," *Res. Eng. Des.*, vol. 31, no. 4, pp. 383–409, Oct. 2020.

[19] K. Kim, S. Cho, and W. Chung, "HD map update for autonomous driving with crowdsourced data," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1895–1901, Apr. 2021.

[20] J. Leng et al., "A loosely-coupled deep reinforcement learning approach for order acceptance decision of mass-individualized printed circuit board manufacturing in Industry 4.0," *J. Cleaner Prod.*, vol. 280, Jan. 2021, Art. no. 124405.

[21] X. Guo, Y. Cao, J. Zhou, Y. Huang, and B. Li, "HDM-RRT: A fast HD-map-guided motion planning algorithm for autonomous driving in the campus environment," *Remote Sens.*, vol. 15, no. 2, p. 487, Jan. 2023.

[22] F. Wu, W. Yang, J. Lu, F. Lyu, J. Ren, and Y. Zhang, "RLSS: A reinforcement learning scheme for HD map data source selection in vehicular NDN," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10777–10791, Jul. 2022.

[23] S. Arshad, M. Sualeh, D. Kim, D. V. Nam, and G.-W. Kim, "Clothoid: An integrated hierarchical framework for autonomous driving in a dynamic urban environment," *Sensors*, vol. 20, no. 18, p. 5053, Sep. 2020.
[24] M. Wu, D. Ye, J. Kang, and R. Yu, "Collaborative data collection with hybrid vehicular crowd sensing in smart cities," in *Proc. 9th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2017, pp. 1–6.
[25] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO series in 2021," 2021, *arXiv:2107.08430*.
[26] X. Lu et al., "Reinforcement learning-based physical cross-layer security and privacy in 6G," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 425–466, 1st Quart., 2023.
[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. vol. 2, no. 1. Cambridge, MA, USA: MIT Press, Nov. 2018.
[28] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[29] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 12, 1999, pp. 1–12.
[30] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Int. Conf. Mach. Learn.*, Jun. 2014, pp. 387–395.
[31] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Conf. Neural Inf. Process. Syst.*, Dec. 2000, pp. 1008–1014.
[32] H. Peng and X. S. Shen, "DDPG-based resource management for MEC/UAV-assisted vehicular networks," in *Proc. IEEE 92nd Veh. Technol. Conf. (VTC-Fall)*, Nov. 2020, pp. 1–6.
[33] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel DDPG method with prioritized experience replay," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 316–321.
[34] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[35] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

Daniel Mawunyo Doe (Member, IEEE) received the B.S. degree in computer engineering from the Kwame Nkrumah University of Science and Technology, Kumasi, Ghana, and the M.S. degree in computer science and engineering from the University of Electronic Science and Technology of China (UESTC). He is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, University of Houston, USA. His general research interests include game theory, blockchains, federated learning, wireless networks, big data, and cloud computing.

Dawei Chen (Member, IEEE) received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2015, and the Ph.D. degree from the University of Houston, Houston, TX, USA, in 2021. After this, he joined Toyota Motor North America, InfoTech Laboratories, where he is currently a Principal Researcher. His research interests include edge/cloud computing, federated learning/analytics, connected vehicles, and wireless networks.

Kyungtae Han (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from The University of Texas at Austin, Austin, TX, USA, in 2006. He is currently a Senior Principal Scientist with the InfoTech Laboratories, Toyota Motor North America, Mountain View, CA, USA. Prior to joining Toyota, he was a Research Scientist with the Intel Laboratories, Santa Clara, CA, USA, and a Director of Locix Inc., San Bruno, CA, USA. His research interests include cybephysical systems, connected and automated vehicle techniques, and intelligent transportation systems.

Haoxin Wang (Member, IEEE) received the B.S. degree in control science and engineering from the Harbin Institute of Technology, China, in 2015, and the Ph.D. degree in electrical and computer engineering from The University of North Carolina at Charlotte in 2020. From 2020 to 2022, he was a Research Scientist with Toyota Motor North America, InfoTech Laboratories. He is currently an Assistant Professor with the Department of Computer Science, Georgia State University, and leads the Advanced Mobility & Augmented Intelligence (AMAI) Laboratory. His current research interests include mobile AR/VR, autonomous driving, digital twins, and edge intelligence.

Jiang Xie (Fellow, IEEE) received the B.E. degree in electrical and computer engineering from Tsinghua University, Beijing, China, the M.Phil. degree in electrical and computer engineering from The Hong Kong University of Science and Technology, and the M.S. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology. She joined the Department of Electrical and Computer Engineering, University of North Carolina at Charlotte (UNC-Charlotte) as an Assistant Professor in August 2004, where she is currently a Full Professor. Her current research interests include resource and mobility management in wireless networks, mobile computing, the Internet of Things, and cloud/edge computing. She received the U.S. National Science Foundation NSF Faculty Early Career Development (CAREER) Award in 2010, the Best Paper Award from IEEE Global Communications Conference in 2017, the Best Paper Award from IEEE/WIC/ACM International Conference on Intelligent Agent Technology in 2010, and the Graduate Teaching Excellence Award from the College of Engineering at UNC-Charlotte in 2007. She is on the editorial boards of IEEE TRANSACTIONS ON WIRELESS NETWORKING, IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING, and *Journal of Network and Computer Applications* (Elsevier).

Zhu Han (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University in 1997 and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively. From 2000 to 2002, he was a Research and Development Engineer with JDSU, Germantown, Maryland. From 2003 to 2006, he was a Research Associate with the University of Maryland. From 2006 to 2008, he was an Assistant Professor with Boise State University, ID, USA. Currently, he is a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department and the Computer Science Department, University of Houston, TX, USA. His main research targets on the novel game-theory related concepts critical to enabling efficient and distributive use of wireless networks with limited resources, wireless resource allocation and management, wireless communications and networking, quantum computing, data science, smart grids, security, and privacy. He received an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the *Journal on Advances in Signal Processing* in 2015, IEEE Leonard G. Abraham Prize in the field of Communications Systems (Best Paper Award in IEEE JSAC) in 2016, and several best paper awards in IEEE conferences. He was an IEEE Communications Society Distinguished Lecturer from 2015 to 2018, has been an AAAS fellow since 2019, and has been a ACM Distinguished Member since 2019. He is also a 1Science. He is also the Winner of the 2021 IEEE Kiyo Tomiyasu Award, for outstanding early to mid-career contributions to technologies holding the promise of innovative applications, with the following citation: "for contributions to game theory and distributed management of autonomous communication networks."