Service Offloading with Deep Q-Network for Digital Twinning Empowered Internet of Vehicles in Edge Computing

Xiaolong Xu, Bowen Shen, Sheng Ding, Gautam Srivstava, Muhammad Bilal, Mohammad R. Khosravi, Varun G Menon, Mian Ahmad Jan, Maoli Wang*

Abstract—With the potential of implementing computing-intensive applications, edge computing is combined with digital twinning (DT) empowered Internet of Vehicles (IoV) to enhance intelligent transportation capabilities. By updating digital twins of vehicles and offloading services to edge computing devices (ECDs), the insufficiency in vehicles' computational resources can be complemented. However, owing to the computational intensity of DT empowered IoV, ECD would overload under excessive service requests, which deteriorates the quality of service (QoS). To address the problem, a multi-user offloading system is analyzed, where the QoS is reflected through the response time of services. Then, a service offloading method with deep reinforcement learning, named SOL, is proposed for DT empowered IoV in edge computing. To obtain optimized offloading decisions, SOL leverages deep Q-network (DQN), which combines the value function approximation of deep learning and reinforcement learning. Eventually, experiments with comparative methods indicate that SOL is effective and adaptable in diverse environments.

Index Terms—Internet of Vehicles; Edge Computing; Digital Twinning; Service Offloading; Deep Reinforcement Learning

1 Introduction

The Internet of Vehicles (IoV) is an evolution of vehicular ad hoc networks (VANETs), where vehicles are equipped with a variety of Internet of Things (IoT) equipments and envisioned as intelligent objects [1]. In the IoV, an intelligent vehicle is capable of V2X (vehicle to everything) communication. Specifically, an intelligent vehicle can share information with other vehicles through V2V (vehicle to vehicle) communications. Rather than observing the condition by a single car, V2V enables a broader view by sharing the traffic information observed by multiple vehicles, which can significantly reduce accidents caused by the blind spot [2]. Meanwhile, intelligent infrastructures like roadside units (RSUs) and smart traffic lights are deployed to analyze the vehicles in a specific region, then provide vehicles with external information through V2I (vehicle to infrastructure) communications [3]. Similarly, V2P (vehicle to pedestrian) communication enables vehicles and pedestrians to deliver commands and safety warnings [4]. With V2X communication in the IoV, intelligent vehicles have the potential to adjust the driving status in time and avoid the occurrence of traffic accidents and enhance the users driving experience.

Further, the digital twinning (DT) technology leverages machine learning and IoT technologies to create digital replicas of physical objects. The replica has its properties cloned from their original versions, and constantly update themselves with real-time data from sensors. Empowered by DT technology, a virtual twin of vehicle in the IoV is generated and mapped to the physical vehicle with IoT technologies [5]. The DT empowered IoV focuses on collecting the state information of the vehicle and surroundings through the smart sensor devices, and sharing the information with surrounding vehicles and infrastructures [6]. With the collected information, the digital twins are updated constantly to keep consistent with the physical vehicles. Then, through the technologies including augmented reality (AR) simulation and artificial intelligence (AI) predictive analytics, vehicles are provided with enhanced intelligence. Comparing with the traditional IoV, DT empowered IoV can easily access to the digital twins of vehicles instead of applying for and integrating numerous external data sources like the surveillance system and the remote sensing

- *X. Xu and B. Shen are with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China. Email: xlxu@ieee.org, bwshen@nuist.edu.cn*
- *X. Xu is also with the Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science and Technology, Nanjing 210044, China and the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China*
- *S. Ding is with the Blockchain Laboratory of Agricultural Vegetables, Weifang University of Science and Technology, Shouguang, Weifang, 262700, China. Email: dingsheng@wfust.edu.cn*
- *G. Srivstava is with the Department of Mathematics and Computer Science, Brandon University, Canada. Email: srivastavag@brandonu.ca*
- *M. Bilal is with the Department of Computer and Electronics Systems Engineering, Hankuk University of Foreign Studies, Yongin-si, Gyeonggi-do, 17035, Korea. Email: m.bilal@ieee.org*
- *M. R. Khosravi is with the Department of Computer Engineering, Persian Gulf University, Bushehr, Iran, and the Department of Electrical and Electronic Engineering, Shiraz University of Technology, Shiraz, Iran. E-mail: mohammadkhosravi@acm.org*
- *V. G Menon is with the Department of Computer Science and Engineering, SCMS School of Engineering and Technology, Ernakulam 683576, India. Email: varunmenon@scmsgroup.org*
- *M. A. Jan. is with the Department of Computer Science, Abdul Wali Khan University, Mardan, Pakistan. E-mail: mianjan@tdtu.edu.vn*
- *M. Wang is with the School of Cyber Science and Engineering, Qufu Normal University, Qufu 273165, China. Email: wangml@qfnu.edu.cn (corresponding author)*

Manuscript received xx xx, 20xx; revised xx xx, 20xx.

(RS) system. Under such circumstances, the data mining, simulation, and analytics of the IoV can be enhanced by DT.

As most of the collected data in the DT empowered IoV are in the raw form (i.e., unprocessed images and videos), they cannot be directly used for control and services [7]. Thus, a powerful computing platform is required to refine the massive collected data, then feedback the extracted instructions to vehicles and passengers [8]. Usually, the processing of vehicular data requires technologies such as object detection and AR, which are computationally intensive operations [9]. To extend intelligent vehicles' capabilities, the cloud and edge computing solutions provide DT empowered IoV with a platform as a service (PaaS) [10]. The data and service requests collected by vehicles are offloaded to the cloud data center through RSU. After data being processed at the cloud infrastructure, the refined data are fed back in the form of instructions or services [11]. Technically, the cloud data center is composed of centralized large-scale computer clusters with high performance. To reduce the cost of construction and facility maintenance, it is usually built in areas far away from end-users. Therefore, service offloading to the cloud will generate high latency during data transmission and is easy to cause bandwidth tension [12]. As a complementary paradigm of cloud computing, edge computing provides appropriate solutions in the DT empowered IoV by offloading service requests to edge computing devices (ECDs), servers deployed close to vehicles and other end-users, for execution and data extraction [13].

Despite the advantages of fast transmission and sufficient bandwidth resources, edge computing has its own challenges. Considering the distributed manner of ECDs, the computing capacity of each independent ECD is smaller than the cloud data center. Thus, the resources in each ECD are supposed to be fully utilized to attain higher efficiency and quality of service (QoS) [14]. Further, the load balancing in ECD is an important issue, and mishandling of service offloading can cause load imbalance. Consequently, some devices in ECDs would underperform due to excessive service requests, and other would be underutilized. To enhance the performance of edge computing and provide reliable services to passengers, an effective service offloading method is needed in the DT empowered IoV [15].

For the dynamic offloading control, deep reinforcement learning (DRL) is adopted to evaluate and choose decisions where the collective utilization is optimized [16]. Among the existing DRL algorithms, the deep Q-network (DQN) has gained attention as a modification of Q-learning, which takes the advantage of temporal-difference learning from reinforcement learning (RL) and the function approximation from deep learning (DL) [17]. In this paper, a dynamic service offloading method, named SOL, is proposed based on DQN in edge computing. Specifically, the contributions of this paper are as follows:

- Analyze the QoS level of DT empowered IoV services in respect of response time in a multi-user offloading system.
- Model the ECD as the agent and formalize the state, action, and reward in DRL to optimize the QoS level of the offloading system.
- Apply DQN with experience replay and target network [17] to solve the problem of DT empowered IoV service offloading in edge computing.
- Conduct comparative experiments with a real-world IoV service dataset to evaluate the effectiveness and adaptability of SOL.

The rest of paper is organized as follows. In section 2, the related work is summarized. In section 3, the model of service offloading in edge computing is described. In section 4, details of DRL and SOL are presented. Then, comparison experiments are conducted in section 5. Finally, in section 6 the achievements of this paper are concluded and future works are discussed.

2 Related Work

So far, various applications in the DT empowered IoV have been proposed to enhance the QoS, safety and security of transportation [18]. However, the generated data of such applications are large in scale and has much redundancies, therefore not suitable for local computing and existing cloud computing paradigms [19]. In [20], Hu et al. addressed the scale-sensitive problem of existing object detection, then modified the deep convolutional neural network for vehicle detection with a large variance of scales to guarantee the accuracy and safety in IoV. From another perspective, Liu et al. [21] exhibited the outstanding performance of edge computing on enhancing the security and QoS of autonomous vehicles, including extending computing capacity and reducing energy consumption.

The placement of ECDs has great impact on overall performance of edge computing. Zhao et al. [22] proposed a ranking-based near-optimal placement algorithm to minimize average access delay through SDN techniques in cloudlets placement. Wang et al. [23] studied the ES placement while considering load balancing as well as access delay and adopted mixed integer programming to find the optimal placement. After ECDs are located, task offloading can be taken into operation. In [24], He et al. gave consideration to users' privacy and system cost in mobile edge computing, and proposed a novel task offloading scheme to enhance user experience. Zhou et al. [25] investigated the task offloading under information asymmetry and uncertainty in vehicular fog computing, and proposed a contract optimization to realize the effective server recruitment.

Owing to higher effectiveness of evolutionary algorithms (EAs), researchers widely adopted EAs as a tool for optimizing the offloading problems in edge computing. Guo et al. [26] comprehensively investigated the computation offloading as a mix integer non-linear programming problem, and designed a computation algorithm based on the genetic algorithm and particle swarm optimization to minimize the energy consumption of the user equipment. However, EAs are usually iterative algorithms that find the global optimal solutions by updating the current solutions continuously. Thus, the dependency on global information and the considerable time complexity during the iteration of generations become significant drawbacks [27]. If EAs are adopted for the offloading of each service, the time overhead in controlling can be unaffordable for the practical implementation of edge computing empowered IoV.

To obtain decentralized and time-efficient control in the IoV, DRL has been adopted in many aspects of the IoV. To achieve high QoS V2V communication, a decentralized resource allocation mechanism based on DRL is designed in [28]. Benefitting from the decentralized manner, DRL can significantly reduce the transmission overhead and the waiting time for global information. Apart from the efficiency, DRL also exhibits the advantage in adaptability. In [29], Liang et al. adopted DRL to studied the automatic determination of traffic signal duration based on the data collected from sensors. In their model, the actions are changes in the duration of a traffic light, and the reward is the difference in cumulative waiting time between two signal cycles. Meanwhile, Zhou et al. [30] proposed a DRL-based car-following model, which can make adjustments in driving behaviors under diverse traffic demands, to improve travel efficiency and safety at signalized intersections in real-time. Generally, DRL is promising in achieving distributed control in the dynamic environment of IoV.

3 System Model and Problem Definition

This section describes the system model and service offloading in edge computing. Table 1 presents the key notations and definitions used in this paper.

TABLE 1
Notations and Definitions

Notations	Definitions
N	The number of RSUs
M	The number of ECDs
K	The number of vehicles
R	The set of RSUs, $R = \{r_1, r_2, \cdots, r_N\}$
E	The set of ECDs, $E = \{e_1, e_2, \cdots, e_M\}$
V	The set of vehicles, $V = \{v_1, v_2, \cdots, v_K\}$
$D(t)$	The data size of services at time t, $D(t) = \{d_1(t), d_2(t), \cdots, d_K(t)\}$
C_e	The coverage of ECD
C_r	The coverage of RSU
$dist$	The distance between two network nodes
RT	The response time of services
S	The QoS level of services

3.1 Framework of Service Offloading for DT Empowered IoV in Edge Computing

In the proposed framework, vehicles are denoted by set $V = \{v_1, v_2, \cdots, v_K\}$. For each vehicle, a digital twin of itself is generated with information of position, speed, vehicle gap, and dashcam videos collected by vehicular sensors and cameras. The raw data and service messages of vehicles can be sent to RSUs, denoted by set $R = \{r_1, r_2, \cdots, r_N\}$. With the constant update, we can assume that the cloning is successful, and the functions of the digital twin keep pace with the entity's. Each vehicle can concurrently request one service at time t, and the data to be processed of each vehicular service is denoted by set $D(t) = \{d_1(t), d_2(t), \cdots, d_K(t)\}$, while $d_i(t) = 0$ indicates that no service is requested by vehicle v_i. For RSUs are usually considered as communicating nodes and not capable of a large scale of computing tasks, ECDs are arranged to some certain districts to process the service requests based on digital twins of vehicles with massive data collected by RSUs. The ECDs are denoted by the set $E = \{e_1, e_2, \cdots, e_M\}$. RSUs can communicate with each other as well as ECDs in their transmission range. Generally, the framework of task offloading in DT empowered IoV with edge computing is shown in Fig. 1.

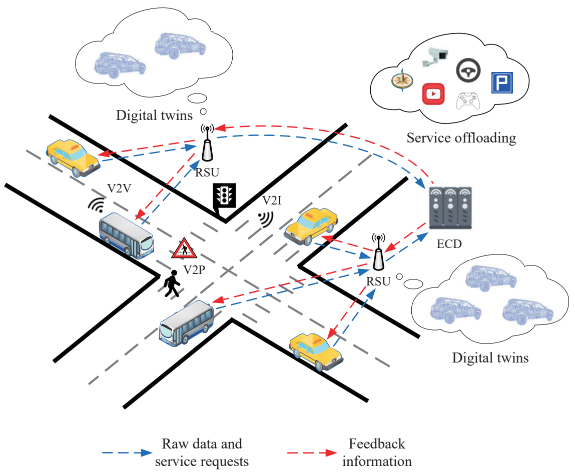

Fig. 1. A framework of service offloading in DT empowered IoV with edge computing.

In the DT empowered IoV, the coverage of each ECD is assumed to be the same and denoted by C_e, while for RSUs, the range is denoted by C_r. Then, every RSU, ECD, and vehicle can be respectively denoted by

$$r_i\left(lat_i, lon_i, C_r\right), \quad 1 \leqslant i \leqslant N, \tag{1}$$

$$e_j\left(lat_j, lon_j, C_e\right), \quad 1 \leqslant j \leqslant M, \tag{2}$$

$$v_k\left(\tilde{lat}_k(t), \tilde{lon}_k(t), d_k(t)\right), \quad 1 \leqslant k \leqslant K, \tag{3}$$

where lat_n and lon_n represent the latitude and longitude of a network node respectively, as the location of vehicle is dynamic with time, $(\tilde{lat}_k(t), \tilde{lon}_k(t), d_i(t))$ is used to represent the state of v_k at time t.

Based on the latitude and longitude, the distance between two nodes (i.e., RSU, ECD or cloud access point) can be calculated by the Euclidean distance as

$$dist\left(node_i, node_j\right) = \sqrt{\left(lat_i - lat_j\right)^2 + \left(lon_i - lon_j\right)^2}. \tag{4}$$

It is guaranteed that the data transmission between a vehicle and an RSU, as well as an RSU and an ECD, is a one-hop transmission. Specifically, each RSU is in the coverage of at least one ECD while each vehicle is in the coverage of at least one RSU as

$$\forall r_i \in R, \ \min_{e_j \in E} dist\left(r_i, e_j\right) \leqslant C_e, \tag{5}$$

$$\forall v_k \in V, \ \min_{r_i \in R} dist\left(v_k(t), r_i\right) \leqslant C_r \tag{6}$$

3.2 QoS Model of DT Empowered IoV Services Offloading in Edge Computing

RSUs in the offloading system can independently choose their computing paradigm in each time period, namely, local computing or edge computing. The response time of a service request can be calculated as the sum of offloading time, execution time and feedback time.

3.2.1 Local Computing Model

When vehicle v_k proposes a service request at time t, and locally executes it, the offloading indicator is $a_k(t) = 0$. In this case, local computing yields a response time of $RT_k^l(t)$, which only includes the execution time of the task by vehicular computing units. The execution time is determined by the processing capacity of resource units and the length of data to be executed. Considering that the processing requirements of vehicular services are usually different, a standard measurement is to divide the vehicular processor into multiple resource units with same local computing capacity of λ_l^{exec}, and u_a of these units are activated for the service. Then the local execution time is calculated as

$$RT_k^l(t) = RT_k^{que}(t) + \frac{f(d_k(t))}{u_a \cdot \lambda_l^{exec}}, \tag{7}$$

where $RT_k^{que}(t)$ is the queuing time of the task, denoted by the difference between the execution starting time and requested time as $RT_k^{que}(t) = T_k^{start} - T_k^{request}$. Meanwhile, $f(d_k(t))$ represents the total computation of the service with the size $d_k(t)$ of raw data.

3.2.2 Offloading Computing Model

When the service of vehicle v_k is determined to be offloaded to ECD, the offloading indicator is $1 \leqslant a_k(t) \leqslant M$, which indicates that the offloading destination is the $a_k(t)$-th ECD in the offloading system. Accordingly, the response time $RT_k^o(t)$ is generated during three parts of offloading computing. First, the data and service request of vehicle are transmitted from v_k to the nearest RSU r_i, and r_i offloads the service to the destination ECD. During this phase, network latency occurs in the data transmission, calculated as

$$\begin{aligned} RT_k^{o,tran}(t) &= RT_v^{o,tran}(t) + RT_r^{o,tran}(t) \\ &= \frac{d_k(t)}{\lambda_v^{tran}} + \frac{d_k(t)}{\lambda_r^{tran}} \end{aligned}, \tag{8}$$

where λ_v^{tran} is the data transmission rate between v_k and r_i while λ_r^{tran} are the data transmission rate between r_i and ECD. According to the Shannon-Hartley theorem, λ_v^{tran} and λ_r^{tran} is affected by the bandwidth B of the channel, signal power p_t, and the average power of the additive white Gaussian noise p_n. As the channel resources of an RSU are often utilized by several vehicles, the bandwidth utilized by each RSU is denoted by $\frac{B}{K_c}$ when K_c vehicles are utilizing the channel concurrently. Thus, λ_v^{tran} is calculated as

$$\lambda_v^{tran} = \frac{B_r}{K_c} \log_2 \left(1 + \frac{p_t}{p_n}\right), \tag{9}$$

analogously, the transmission rate λ_r^{tran} between the ECD and one of N_c RSUs is calculated as

$$\lambda_r^{tran} = \frac{B_e}{N_c} \log_2 \left(1 + \frac{p'_t}{p'_n}\right) \tag{10}$$

After the service and digital twin data of v_k being offloaded, the destination ECD will take time for execution. Analogous to (7), the execution time of ECD is calculated as

$$RT_k^{o,exec}(t) = RT_k^{que}(t) + \frac{f(d_k(t))}{u_a \cdot \lambda_o^{exec}}. \tag{11}$$

where λ_o^{exec} represents the execution capacity of the ECD, usually considered as $\lambda_o^{exec} = n \cdot \lambda_l^{exec}$.

After the task is executed, the computing results are report back to the RSU to update the digital twin and give instruction to the vehicle. Usually, the feedback data is condensed with a relatively small size of d'_k. Thus, the feedback time during feedback is considered negligible.

Based on (8) and (11), the total response time of the service proposed by v_k at time t by offloading computing is $RT_k^o(t) = RT_k^{o,tran}(t) + RT_k^{o,exec}(t)$

3.2.3 QoS Measurement

To quantify and measure the QoS, the maximum tolerable response time RT_{th} is used as a standard to normalize the indicator of QoS. The QoS level of response time in local computing and offloading computing are calculated as

$$S_k^l(t) = 1 - \frac{RT_k^l(t)}{RT_{th}}, \tag{12}$$

$$S_k^o(t) = 1 - \frac{RT_k^o(t)}{RT_{th}}. \tag{13}$$

3.3 Problem Definition

In the multi-user offloading system, the goal is to maximum the average QoS level of vehicular services through an optimal offloading strategies set $A(t) = \{a_1(t), a_2(t), \cdots, a_K(t)\}$ at each time period t. Based on the models given above, the problem of service offloading in DT empowered IoV is formulated as

$$\max_{A(t)} \sum_{k=1}^{K} \left[S_k^l(t) + \sum_{m=1}^{M} S_k^o(t) \Pr\left[a_k(t) = m\right] \right] / \sum_{k=1}^{K} \mathrm{Sgn}(d_i(t)), \tag{14}$$

$$s.t. \quad \forall v_k \in V,\ a_k(t) \in [0, M], \tag{15}$$

$$\forall v_k \in V,\ S_k^o(t) \geqslant 0,\ S_k^l(t) \geqslant 0, \tag{16}$$

where $\Pr\left[a_k(t) = m\right]$ is the probability of $a_k(t) = m$, i.e., the value is 1 if $a_k(t) = m$, otherwise, 0. Meanwhile, $\mathrm{Sgn}(d_i(t))$ is the sign of $d_i(t)$, i.e., $\mathrm{Sgn}(d_i(t)) = 1$ indicates that $d_i(t)$ is positive, and when $d_i(t) = 0$, $\mathrm{Sgn}(d_i(t)) = 0$. As an element of A, $a_k(t)$ represents the offloading destination, subject to constraint (15). When $a_k(t) = 0$, the service will be locally executed. Otherwise, it will be offloaded to the corresponding ECD for execution. Meanwhile, (16) indicates that the QoS is not negative, i.e., the service response time should be within the maximum tolerable time.

4 SOL for DT Empowered IoV Services Offloading

In this section, SOL is designed for the service offloading in edge computing enabled IoV. First, the framework of RL is introduced in service offloading. Then, the drawback of a primitive RL algorithm named Q-learning is analyzed, and a DRL algorithm named DQN is leveraged for SOL.

4.1 Framework of Reinforcement Learning in SOL

RL is one of the significant branches of machine learning alongside supervised learning and unsupervised learning. It refers to the process of achieving the highest cumulative rewards through the exploration of the environment and the exploitation of previous knowledge. During such trial-and-error process, the agent in RL can obtain the perception of the environment and the decision-making strategy.

In the offloading system, the ECD is enabled the controlling of offloading decisions and viewed as the agent in RL. There are three key elements of an agent, namely the state (s), the action (a), and the reward (R). Usually, the state is also considered the environment that the agent reacts to. In SOL, the state consists of two components, the available units of ECD, and the average QoS level of each vehicle in the offloading system calculated as equation 14. When the ECD receives a service request, it searches for an optimal action $a_k(t)$ available in its current state. Based on the action indicator $a_k(t)$, the ECD decides where to offload and execute the service request. After making offloading decision and execution, the QoS level of service is evaluated in terms of the vehicle's response time as $S_k(t)$, then fed back to ECD as the reward. In general, the goal of RL is to obtain the highest cumulative reward in a learning episode.

Among the RL algorithms, Q-learning has proved to be effective in model-free learning problems [31]. In Q-learning, the agent is given a Q-table which records the Q-value (i.e., quality) of each pair of state and action as $Q(s, a)$. For each step, the agent selects an action a_t at the state s_t which brings it the highest reward, then calculates and updates $Q(s_t, a_t)$ based on the action it chooses and the reward it gets as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot \delta_t, \tag{17}$$

where α is the learning rate parameter that satisfies $0 \leqslant \alpha \leqslant 1$ and determines the extent to which the newly acquired knowledge overrides the old knowledge. Meanwhile, δ_t is the difference between the actual value of $Q(s_t, a_t)$ and the estimated value of it through the Q-table, calculated as

$$\delta_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t), \tag{18}$$

where γ represents the discount factor of future reward, s_{t+1} is the next state after the agent performing a_t , and r_t is the instant reward experienced by the agent, also denoted as the QoS level of service. Notice that, if the response time exceeds the maximum tolerable time, the reward r_t is set as $r_t \leftarrow \min(r_t, 0)$ automatically as a punishment. Specifically, the discount factor satisfies $0 \leqslant \gamma \leqslant 1$, and the larger γ means that the agent has a clearer view toward the future while lower γ means that the agent is more focused on the instant reward. Usually, Q-learning starts with a lower discount factor and increases it towards its final value to accelerates learning.

As directly choosing the action with maximal Q-value encourages exploitation but lacks exploration, agents might fall into the local optimum. Thus, a certain degree of randomness is allowed by introducing the ϵ-greedy in strategy selection. Specifically, agents select the strategy with the highest Q-value with probability $\Pr\left[s_i(t) = s_{best}\right] = 1 - \epsilon$ to exploit knowledge, while with probability ϵ, they randomly select another action to explore for more available choice. Usually, ϵ decreases over time to encourage exploration during the early phase and limit the blindness and fluctuation of agents' decision-making in the later phase.

4.2 SOL with Deep Q-Network

The primitive reinforcement learning method has a significant disadvantage that it requires a Q-table to store the Q-values of all possible state-action pairs. However, the number of states is large or even infinite, the traverse and update of Q-table become time-consuming. Moreover, there exist many state-action pairs that are similar but not identical in a complex Q-table. Therefore, the traditional Q-learning method will become ineffective since the possibility of the agent to access a specific state-action pair is relatively small. To tackle the problem, a practical approach is to approximate the Q-values of different state-action pairs with DNN, which leads to the primary essence of DQN [17]. Intuitively, the differences between Q-learning and DQN in offloading decision making are shown in Fig. 2.

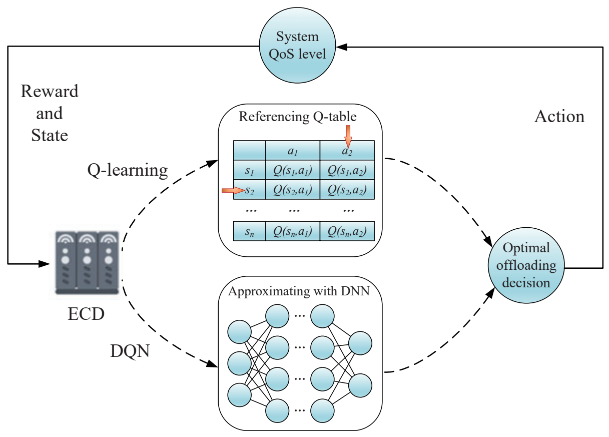

Fig. 2. Differences between offloading decision making based on Q-learning and DQN.

Practically, the proposal of DQN successfully combined RL with DL while tackling the challenges in the inconsistency between them. Usually, DL assumes that the distribution of data samples is in an independent manner. However, the states and actions in RL are usually highly correlated, which is not consistent with the requirement of DL. To mitigate the correlation in data, a technique of experience replay is introduced. Technically, a structure of experience pool D, which stores the experience of each step as $e_t\left(s_t, a_t, r_t, s_{t+1}\right)$, is adopted to enable experience replay in DQN. During the network training, a minibatch of the experience is randomly drawn from D for training such that the distribution of data can be averaged, and the correlations can be alleviated.

Another feature of DQN is to generate a target Q value in a separate network (i.e., the target network Q^{tar}). Unlike the original network (i.e., the prediction network Q^{pre}) which updates the parameters θ in every iteration, θ^- in the target network are only periodically updated in every C iterations and stay fixed in other steps. Specifically, after C rounds of updates by the prediction network, the target network is updated by a copy of the prediction network.

This feature adds a delay between the update of the network and the effect on the targets y_j and further stabilizes the performance of DQN.

With DNN, the Q-value of state-action pair (s_t, a) is estimate as $Q^{pre}(s_t, a; \theta) \approx Q(s_t, a)$, where the parameter θ is a vector of weights in the DNN. To evaluate the accuracy of the approximation and further train the network, the loss function is introduced as

$$L_i(\theta_i) = \mathbb{E}\left[(y_i - Q^{pre}(s, a; \theta_i))^2\right], \tag{19}$$

where y_i represents the target Q-value generated by the target network of

$$y_i = r + \gamma \max_{a'} Q^{tar}\left(s_{t+1}, a', \theta_i^-\right). \tag{20}$$

By minimizing $L_i(\theta_i)$ through updating weight θ repeatedly, the network can be trained to be more accurate. Technically, minibatch stochastic gradient descent (MSGD) is applied to minimize the difference between the output of the target network and the prediction network. More precisely, the pseudo code of DQN is shown in Algorithm 1.

Algorithm 1 SOL with Deep Q-Network

```
1:  Initialize experience pool $D$ with the size of $N$
2:  Initialize $Q^{pre}$ and $Q^{tar}$ with same random weights $\theta$
3:  for $episode = 1$ to $M$ do
4:      for $t = 1$ to $T$ do
5:          Approximate Q-values of all actions at state $s$
6:          Select the optimal offloading decision $a_t$ based on $\epsilon$-greedy policy
7:          Perform service offloading or local computing according to $a_t$
8:          Calculate the reward $r_t$ and the next state $s_{t+1}$
9:          Store experience $e_t(s_t, a_t, r_t, s_{t+1})$ in $D$
10:         Perform MSGD to update the parameters $\theta$ of prediction network $Q^{pre}$ through minimizing $L(\theta)$
11:         Update the target network $Q^{tar}$ every $C$ steps
12:     end for
13: end for
```

4.3 SOL Review

Generally, SOL is designed on the logical basis shown in Fig. 3. The basic idea of SOL is to enable the ECD to make optimal offloading decisions through RL. With the exploration of the unknown environment, the agent in RL can learn from the feedback reward. Meanwhile, the exploitation of experienced knowledge enables the agent to select optimal action at each state, jointly considering the instant reward and long-term reward. However, as the environment of the IoV service offloading system is dynamic and sophisticated, the space of states can be vast or infinite. If primitive RL algorithms like Q-learning are adopted, the update and search for optimal offloading decisions generate a significant overhead of storage and time. Moreover, the similar but not identical states significantly increase the agent's exploration range and will lead to slow convergence of RL. To reduce the overhead in storage and time while fastening convergence, a DRL algorithm named DQN is adopted in SOL. Instead of referencing the Q-table to find the optimal decision, DQN introduced the function value approximation of DL to estimate the Q-value of state-action pairs. Also, with the features of experience replay and target network, DQN successfully alleviates the inconsistency between RL and DL, and can achieve satisfying performance in SOL.

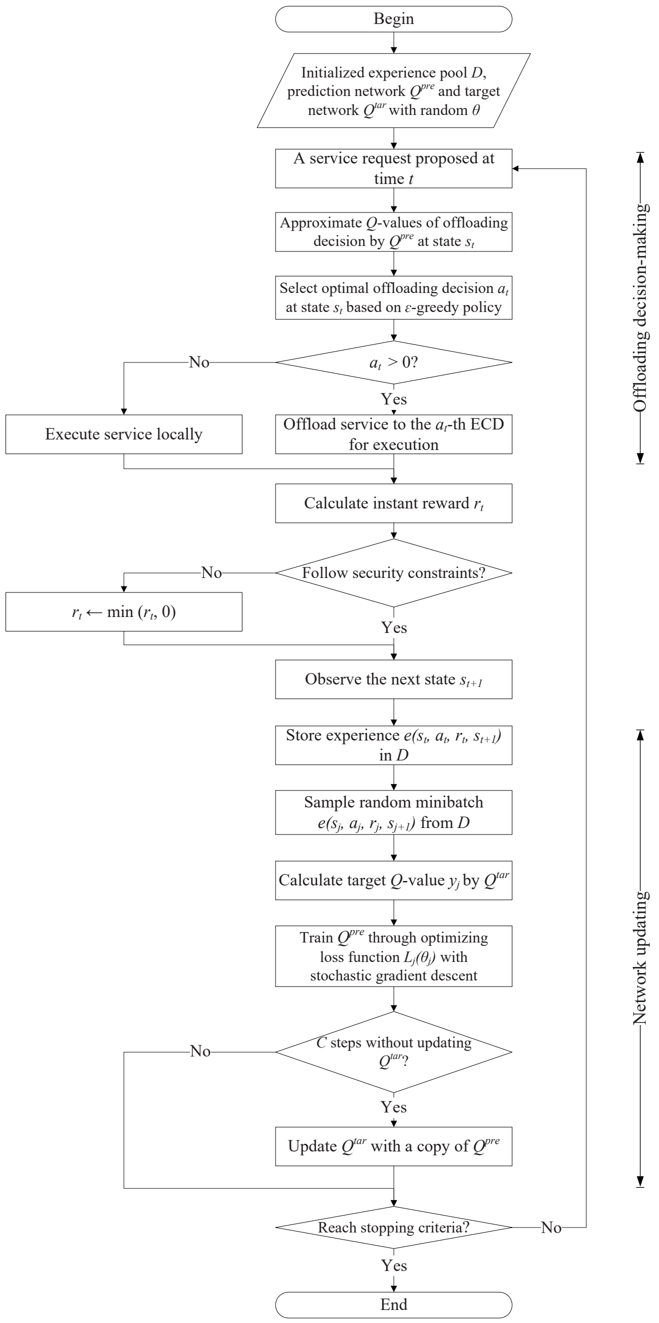
Begin
Initialized experience pool D, prediction network Q^{pre} and target network Q^{tar} with random θ
A service request proposed at time t
Approximate Q-values of offloading decision by Q^{pre} at state s_t
Select optimal offloading decision a_t at state s_t based on ε-greedy policy
$a_t > 0$?
No
Yes
Execute service locally
Offload service to the a_t-th ECD for execution
Calculate instant reward r_t
Follow security constraints?
No
Yes
$r_t \leftarrow \min(r_t, 0)$
Observe the next state s_{t+1}
Store experience $e(s_t, a_t, r_t, s_{t+1})$ in D
Sample random minibatch $e(s_j, a_j, r_j, s_{j+1})$ from D
Calculate target Q-value y_j by Q^{tar}
Train Q^{pre} through optimizing loss function $L_j(\theta_j)$ with stochastic gradient descent
C steps without updating Q^{tar}?
No
Yes
Update Q^{tar} with a copy of Q^{pre}
Reach stopping criteria?
No
Yes
End
Offloading decision-making
Network updating

Fig. 3. The programming flowchart of SOL.

5 Experimental Evaluation

In this section, SOL is implemented and experiments are conducted based on the real-world IoV service requests. Then, comparative offloading strategies are introduced. Finally, the results of SOL and comparative offloading strategies under different circumstances are presented, and the effectiveness and adaptability of SOL are verified based on the experimental results.

5.1 Experiment Setup

Two real datasets of IoV service requests in Nanjing are applied in the experiment. One dataset contains details of 436 activated RSUs in Nanjing, including their latitude and longitude values. Based on the RSU locations, partitioning around medoids (PAM) clustering is adopted with the parameter $K = 40$ to simulate the placement of ECDs and the assignment of RSUs. As shown in Fig. 4, on part of the brief road map of Nanjing, the RSUs and ECDs in one cell of the offloading system are marked with blue dots and red server icons respectively. The 3 ECDs and 26 RSUs (including 3 co-located with each ECD) are analyzed in the experiments.

The other dataset contains vehicular service requests collected by RSUs in 30 consecutive days (from 00:00:00 Sep. 1st to 23:59:59 Sep. 29th). The total number of service requests is more than 160 million. From the second dataset, the service requests in one cell of the offloading system are extracted for comparative analysis.

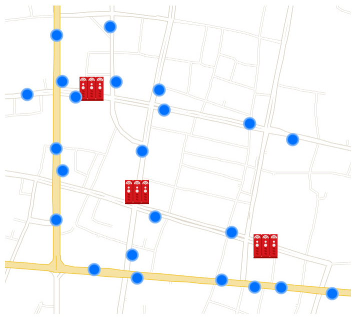
Fig. 4. Distribution of RSUs and ECDs in an offloading system.

5.2 Comparative Offloading Strategies

5.2.1 *Entirely Local Computing*

Entirely local computing is a conventional paradigm which depends only on the vehicles' local execution capacity. Entirely local computing requires no additional controlling strategy, and is used as a baseline to evaluate the optimization capability of other offloading strategies.

5.2.2 *Nearest Neighbor Offloading Computing*

Contrary to entirely local computing, nearest neighbor offloading strategy enables all the service requests and raw data to be offloaded to the nearest ECD for execution. As the location of RSUs and ECDs are fixed, the nearest ECD of each RSU can be determined. When the computational resources of ECD are abundant, this strategy can achieve a high level of QoS without complicated controlling. However, as the local computing units are not utilized, and the distribution of workload is uneven, excessive offloaded services will increase the risk of ECD being overloaded and severely lower the QoS level of the offloading system.

5.2.3 *First Fit Offloading Computing*

First fit is an online algorithm where the service is offloaded to the nearest ECD that can accommodate it. When first fit algorithm begins, it searches for the closest ECD to the RSU which collected a service request. If the ECD has insufficient idle resource units for the service, it will be offloaded to the next closest ECD with sufficient resources. If no ECD is capable, the service will be executed by the computing devices of the vehicle which proposes the request.

5.3 Analysis on the Adaptability of Offloading Strategy

As the real condition of IoV services in cities are various, e.g., the number of vehicles and ECDs varies with the development of cities. Thus, the offloading strategy needs to be adaptive so that it can be applied widely. To verify the adaptability of SOL, four sets of controlled experiments with diversity in services conditions are conducted, and the performance of SOL is evaluated.

The controlled value of variables in the comparative analysis are listed in Table 2. In each set of experiment, there is one variable with its value flucuates around the controlled value and the others remain unchanged.

TABLE 2
Controlled Variable Settings

Variable description	Controlled value
ECD execution capacity	$5 \times$ local execution capacity
Number of ECD	3
Number of service requests	5 per vehicle
Average size of raw data	50 MiB per request

5.3.1 *Analysis on the Variety of ECD Execution Capacity*

Experiments are conducted with different ECD's execution capacity, and the results are shown in Fig. 5. In this set of experiments, the ratio of ECD execution capacity to local execution capacity ranges from 3 to 7. As the results indicate, SOL outperforms entirely local computing, nearest neighbor offloading, and first fit offloading in response time. When the capacity of ECD is insufficient, the risk of ECD being overloaded is high if no effective offloading strategy is adopted. Thus, the QoS level of vehicular services by nearest neighbor offloading is severely reduced by long response time. In contrast, when the execution capacity of ECD is ample, the difference in response time between SOL and the other offloading strategies is small. As the ECDs can efficiently execute most of the services, offloading computing is usually the optimal choice.

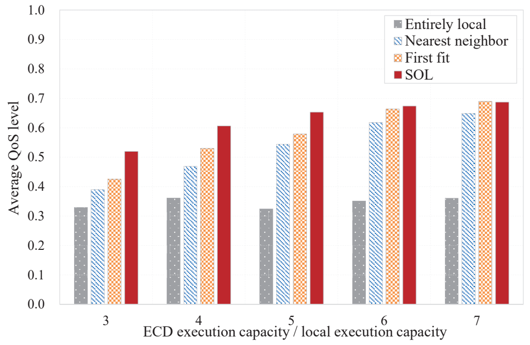

Fig. 5. Comparison of QoS level with variety in ECD capacity.

5.3.2 Analysis on the Variety of ECD Number

When the number of ECDs in the offloading system are different, the QoS level of vehicular services are shown in Fig. 6. With other variables unchanged, the number of ECDs ranges from 1 to 5 in this set of experiments. The QoS level of SOL is generally the highest despite the little disadvantage over first fit when ECD number is 5. When ECDs are sparsely deployed, the ECDs can be easily overloaded by the excessive service requests. Thus, SOL tends to assign the services to be executed locally and has a slight advantage over other strategies. In contrast, when ECDs are ample, the service requests and the workload of ECDs are more balanced with SOL or first fit offloading strategy, and overloading is unlikely to occur during offloading computing.

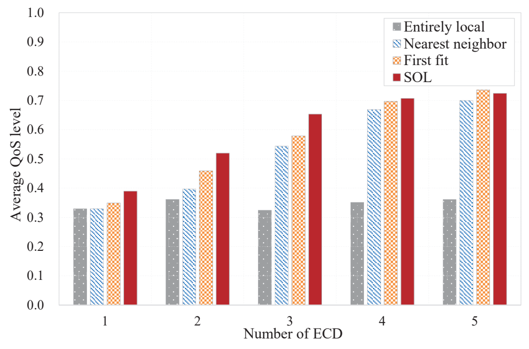

Fig. 6. Comparison of QoS level with variety in ECD number.

5.3.3 Analysis on the Variety of Service Number per Vehicle

Fig. 7 illustrates the impact on the QoS level by the number of services per vehicle. In this set of experiments, we assume each vehicle can propose multiple service requests at different time, and the number of proposed service requests per vehicle ranges from 3 to 7, while other variables remain unchanged. The QoS level of response time by offloading method goes down as the number of services rises, while SOL keeps the decline smaller than first fit and nearest neighbor offloading. The advantage of SOL is that ECD selectively executes some of the services while others are executed locally, which reduces the latency in queuing. When the execution capacity of ECD goes beyond the service requests of vehicles, the QoS level of first fit and nearest neighbor offloading is close to the one of SOL and both outperform local computing. In addition, as the bandwidth of ECD is usually considered fixed, the intensive data transmission also has an impact on the offloading time when the communication is frequently.

5.3.4 Analysis on the Variety of Average Size of Raw Data

In Fig. 8, the QoS level with diversity in the average size of raw data is analyzed. Experiments are conducted with the average size of raw data ranging from 30MiB to 70MiB, while the other variables remain unchanged. It is intuitive that the QoS level declines with the rise in the size of raw data. As the computing capacity of on-board devices is usually insufficient, the response time of local computing is intolerable. Simultaneously, the QoS level of nearest neighbor offloading, first fit offloading and SOL also experience

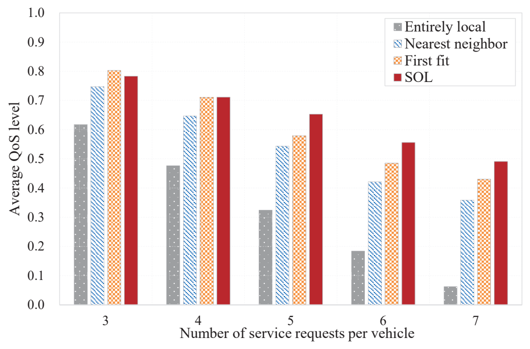

Fig. 7. Comparison of QoS level with variety in service number.

a drop. However, as the execution rate of ECD is much higher than on-board devices, the increase in response time by offloading methods is not significant. Instead, the time overhead generated in data transmission has an impact on the QoS level. Hopefully, 5g communication is promising in mitigating the data transmission time and further enhance the QoS level of service offloading by SOL.

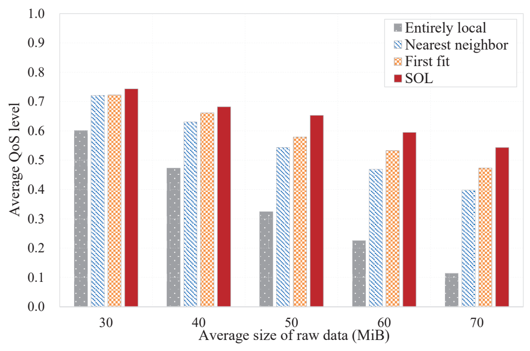

Fig. 8. Comparison of QoS level with variety in average size of raw data.

6 Conclusion

In this paper, edge computing was adopted in the DT empowered IoV to provide vehicular services with a high QoS level, and a service offloading method with deep reinforcement learning named SOL was proposed. First, a multi-user offloading system in DT empowered IoV was modeled with consideration of response time. Then, DQN with experience replay and target network, which exerts the advantages of both RL and DL, is adopted in the offloading system to obtain optimal offloading strategy. The experiments were conducted with a real-world dataset of RSU locations and IoV service requests, and the results verified the effectiveness and adaptability of SOL.

To simplify the model, the IoV service offloading was modeled as a binary offloading process where the services are assumed atomic, i.e., services cannot be divided and executed on more than one devices. In future works, partial offloading can be taken into consideration where a service can be divided into several procedures and offloaded to different ECDs. In this case, computational resources can be better utilized. However, if partial offloading is adopted, the partibility, dependency and priority in the procedures of services need to be thoroughly analyzed, and the offloading decisions are required a strict graph dependency constraint.

ACKNOWLEDGMENT

This research is supported by the Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps under grant no. 2020DB005, NUIST Students' Platform for Innovation and Entrepreneurship Training Program under grant no. 202010300024Z, and the National Natural Science Foundation of China under grant no.61702277.

REFERENCES

[1] J. Contreras-Castillo, S. Zeadally, and J. A. Guerrero-Ibañez, "Internet of vehicles: architecture, protocols, and security," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3701–3709, 2017.

[2] K. Asano, N. Enami, T. Kamada, and C. Ohta, "Person reidentification for detection of pedestrians in blind spots through v2v communications," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 764–770.

[3] L. Chen and C. Englund, "Cooperative intersection management: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 570–586, 2016.

[4] X. Wang, Z. Ning, X. Hu, L. Wang, B. Hu, J. Cheng, and V. C. M. Leung, "Optimizing content dissemination for real-time traffic management in large-scale internet of vehicle systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1093–1105, 2019.

[5] O. Veledar, V. Damjanovic-Behrendt, and G. Macher, "Digital twins for dependability improvement of autonomous driving," in *Systems, Software and Services Process Improvement*, 2019, pp. 415–426.

[6] R. Minerva, G. M. Lee, and N. Crespi, "Digital twin in the iot context: A survey on technical features, scenarios, and architectural models," *Proceedings of the IEEE*, pp. 1–40, 2020.

[7] X. Wang, L. T. Yang, L. Song, H. Wang, L. Ren, and J. Deen, "A tensor-based multi-attributes visual feature recognition method for industrial intelligence," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020, doi: 10.1109/TII.2020.2999901.

[8] M. Zhang, C. Chen, T. Wo, T. Xie, M. Z. A. Bhuiyan, and X. Lin, "Safedrive: online driving anomaly detection from large-scale vehicle data," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2087–2096, 2017.

[9] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 383–398, 2018.

[10] K. Djemame, R. Bosch, R. Kavanagh, P. Alvarez, J. Ejarque, J. Guitart, and L. Blasi, "Paas-iaas inter-layer adaptation in an energy-aware cloud environment," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 127–139, 2017.

[11] M. Abbasi, M. Rafiee, M. R. Khosravi, A. Jolfaei, V. G. Menon, and J. M. Koushyar, "An efficient parallel genetic algorithm solution for vehicle routing problem in cloud implementation of the intelligent transportation systems," *Journal of Cloud Computing*, vol. 9, no. 1, p. 6, 2020.

[12] V. G. Menon, S. Jacob, S. Joseph, and A. O. Almagrabi, "Sdn-powered humanoid with edge computing for assisting paralyzed patients," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5874–5881, 2020.

[13] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.

[14] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2019.

[15] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[16] Z. Ning, P. Dong, X. Wang, M. S. Obaidat, X. Hu, L. Guo, Y. Guo, J. Huang, B. Hu, and Y. Li, "When deep reinforcement learning meets 5g vehicular networks: A distributed offloading framework for traffic big data," *IEEE Transactions on Industrial Informatics*, 2019.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[18] Z. Wang, X. Liao, X. Zhao, K. Han, P. Tiwari, M. J. Barth, and G. Wu, "A digital twin paradigm: Vehicle-to-cloud based advanced driver assistance systems," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–6.

[19] X. Wang, L. T. Yang, Y. Wang, L. Ren, and M. J. Deen, "Adtt: A highly-efficient distributed tensor-train decomposition method for iiot big data," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020, doi: 10.1109/TII.2020.2967768.

[20] X. Hu, X. Xu, Y. Xiao, H. Chen, S. He, J. Qin, and P.-A. Heng, "Sinet: A scale-insensitive convolutional neural network for fast vehicle detection," *IEEE transactions on intelligent transportation systems*, vol. 20, no. 3, pp. 1010–1019, 2018.

[21] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[22] L. Zhao, W. Sun, Y. Shi, and J. Liu, "Optimal placement of cloudlets for access delay minimization in sdn-based internet of things networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1334–1344, 2018.

[23] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 160–168, 2019.

[24] X. He, R. Jin, and H. Dai, "Peace: Privacy-preserving and cost-efficient task offloading for mobile-edge computing," *IEEE Transactions on Wireless Communications*, 2019.

[25] Z. Zhou, H. Liao, X. Zhao, B. Ai, and M. Guizani, "Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8322–8335, 2019.

[26] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2651–2664, 2018.

[27] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, 2016, pp. 261–265.

[28] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for v2v communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.

[29] X. Liang, X. Du, G. Wang, and Z. Han, "A deep reinforcement learning network for traffic light cycle control," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1243–1253, 2019.

[30] M. Zhou, Y. Yu, and X. Qu, "Development of an efficient driving strategy for connected and automated vehicles at signalized intersections: A reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[31] N. Kumar, S. N. Swain, and C. Siva Ram Murthy, "A novel distributed q-learning based resource reservation framework for facilitating d2d content access requests in lte-a networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 718–731, 2018.