Received August 13, 2021, accepted September 5, 2021, date of publication September 10, 2021, date of current version September 29, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3112077*

# Task Offloading and Serving Handover of Vehicular Edge Computing Networks Based on Trajectory Prediction

**BAIQUAN LV, CHAO YANG, XIN CHEN, ZHIHUA YAO, AND JUNJIE YANG**
School of Automation, Guangdong University of Technology, Guangzhou 510006, China
Guangdong Key Laboratory of IoT Information Technology, Guangzhou 510006, China

Corresponding author: Chao Yang (yangchaoscut@aliyun.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 62003094, Grant 62003099, Grant 62003101, and Grant U191140003; in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515011377 and Grant 2019A1515011114; and in part by Guangzhou Science and Technology Plan Project under Grant 202102020573.

**ABSTRACT** Vehicular edge computing (VEC) has emerged as a promising paradigm to ensure the real-time task processing caused by the emerging 5G or high level intelligent assisted driving applications. The computing tasks can be processed via the edge services deployed as the roadside units (RSUs) or moving vehicles. However, the high dynamic topology of the vehicular communication system and the time-varying available computing resources in RSUs make a challenge of the efficient task offloading of vehicles. In this paper, we consider an efficient task offloading scheme for VEC networks based on trajectory prediction, we focus on the serving handover between the adjacent RSUs. The moving vehicles can cooperate with RSUs or the surrounding vehicles for task processing. To reduce the latency of task transmission between vehicles, we present a cooperative vehicle selection method based on trajectory prediction. Then, we propose an efficient task offloading scheme based on deep reinforcement learning (DRL), while the dynamically available computing and communication resources are considered jointly. The simulation results show that the proposed task offloading scheme has great advantages in improving the utility of vehicles.

**INDEX TERMS** Internet of vehicles, edge computing, trajectory prediction, task offloading, deep reinforcement learning.

## I. INTRODUCTION

With the rapid development of 5G wireless communication and intelligent transportation technologies, various emerging applications (such as autonomous driving, augmented reality, face recognition, etc.) appear in recent years [1]–[3]. However, the resources of smart devices are extremely limited, the computing, storage and battery capacity cannot meet the quality of services (QoS) requirements of users. Generally, such applications can be transmitted to remote cloud server [4]. To coper with the service delay caused by the long transmission distance and disorder task offloading schemes, mobile edge computing (MEC) has emerged as a promising approach, the delay and computing sensitive tasks can be processed at the proximate wireless access edge nodes [5], [6].

In the intelligent transportation system (ITS), the traffic conditions on road are extremely complex, which makes the network topology of the internet of vehicles (IoV) change rapidly. The communication between vehicles is called vehicle-to-everything (V2X), mainly including vehicle-to-RSU (V2R) and vehicle-to-vehicle (V2V). The intelligent connected vehicles in IoV have to process a large amount of sensor data in real-time for driving. In the VEC networks, deploying edge servers on both sides of the road or utilizing the available computing resources of surrounding vehicles, the task processing delay can be reduced efficiently [7]. Generally, the moving vehicles on road can obtain a higher QoS by offloading computing tasks to the MEC servers deployed at the RSUs. However, due to the construction cost, the deployed servers at RSUs have limited computing resources. When multiple tasks of multiple vehicles are offloaded to the same server disorderly, the edge

The associate editor coordinating the review of this manuscript and approving it for publication was Yan Huo.

server cannot process such a large amount of data, which may cause the overload, the task calculation efficiency will be affected [8]. Moreover, if the task is switched to the adjacent edge servers that are idle or have redundant resources, it will help reduce the task processing delay, but there will cause the server switching cost of data transmission.

In addition to the task offloading optimization, vehicles can choose a surrounding vehicle, it will obtain the tasks via V2V and perform the task processing [9]. Via the help of surrounding vehicles, the server switching cost can be avoided. However, it is difficult to choose cooperative vehicles under the dynamically changing traffic conditions. When the vehicles and surrounding vehicles belong to a same company (i.e., taxi), or the surrounding vehicles are the public transportation system, the RSUs can obtain the history moving trajectory of the surrounding vehicles. With the development of machine learning (ML) methods, we can use the appropriate ML methods to predict vehicle trajectories [10], [11], and select a reliable cooperative vehicle node based on a given criteria.

In the VEC networks, task offloading schemes become the key role for the users' QoS improving. However, many existing MEC offloading schemes only consider a single transmission scheme, such as: V2R [6], [12] or V2V [13], [14] communications. When multiple tasks need to be processed in VEC networks, the task delay requirements may not be satisfied. It is a big challenge that how to carry out the task offloading while the computing resources in both the RSUs and surrounding vehicles are considered.

In this paper, we propose an optimal task offloading scheme based on the vehicle trajectory prediction, the highly dynamic vehicular topology, vehicular tasks offloading targets and switching judgments of MEC servers deployed as the adjacent RSUs are considered jointly. In order to overcome the server overload in the RSUs and the switching cost, the vehicles can offload part/all of the tasks to the surrounding vehicles. We propose an efficient cooperative vehicle selection method firstly. Specifically, by analyzing the historical driving data collected by vehicles, an accurate prediction of vehicular trajectory in a short time can be obtained. The cooperative vehicle can be selected based on the prediction results. Then, we propose an optimal task offloading scheme while the both the varying V2V and V2R communication links and the varying available computing resources in RSUs and surrounding vehicles are considered. The main contributions of this paper are as follows:

- We propose a comprehensive vehicular task offloading scheme in VEC networks based on trajectory prediction, focusing on serving handover between RSUs and selection of cooperative vehicles.
- Aiming at the dynamical IoV topology, we propose an efficient cooperative vehicle selection method. The LightGBM model is used to predict the vehicle moving trajectory. Then, the most suitable cooperative vehicle is selected by evaluating the set of cooperative vehicles, to make active task offloading scheduling effectively.
- For the task offloading process, we propose an optimization problem to maximize the total utility of the VEC system. We establish a Markov decision process, and an optimal deep Q-network-based task offloading (DQN-based TO) scheme in handover area is proposed.

The rest of the paper is organized as follows. In Section II, we review the related work. In Section III, we propose a VEC system model. A ML based vehicle trajectory prediction scheme is described in Section IV. In Section V, we establish a target function to minimize the energy consumption. The DQN-based task offloading scheme is described in Section VI. we provide numerical experiments about the study in Section VII. Finally, we conclude our work in Section VIII.

## II. RELATED WORK

In recent years, MEC has attracted widespread attention, the DRL [15], [16], DDPG [17], convex optimization [18], [19] algorithms are introduced to determine when/where/how to perform task offloading. DRL can be used to find the online offloading schemes. To maximize the weighted sum calculation rates in the wireless charging MEC system, the authors in [15] designed an online offloading algorithm called DROO based on DRL. In [16], the authors proposed a blockchain-empowered MEC model, and designed a model-free DRL algorithm for online computing offloading to maximize the long-term system utility. When the variables constitute a continuous state space, DDPG or A3C algorithms can be used. Reference [17] proposed a DDPG-based hybrid MEC offloading algorithm to balance each users power consumption in data transmission and task computing. In [19], the authors designed an offloading mechanism of MEC system based on content prediction, including a content prediction model based on Long Short Term Memory (LSTM) and a task offloading strategy based on cross-entropy (CE) method, to maximize the total system throughput. In [18], a partial computation offloading model with a joint communication and computation resource allocation problem was proposed, the convex optimization theory is leveraged to minimize the weighted-sum delay of the offloading model. In [20], the authors proposed an software defined network architecture for spectrum sharing in hybrid satellite-terrestrial networks and an auction-based mechanism to assist the traffic offloading negotiation. Ref. [21] proposed a contract-based traffic offloading and resource allocation mechanism to motivate SBS to choose good contracts in the SDWNs of HetUDNs. However, for the highly dynamic network topology and the extremely unstable vehicular communications, the existing MEC technologies cannot be applied to the IoV directly.

Particularly, there are many applications of MEC-based IoV. To improve the QoS of users in IoV, the authors in [22] designed a DQN model to learn the scheduling of task offloading. In [23], the authors proposed a VEC task scheduling model, in which RSUs can dynamically switch between sleep and working states to save energy consumption. In [24],

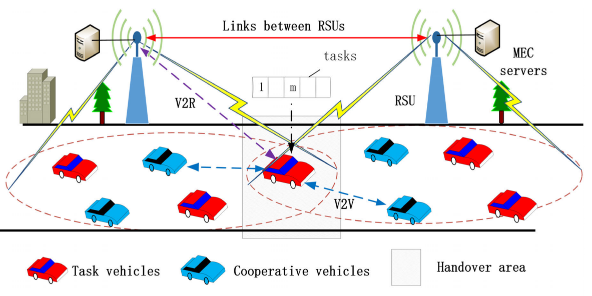


**FIGURE 1. Multi-task offloading scheme between two adjacent RSUs in VEC.**

a two-level VEC architecture was designed to coordinate content sharing between vehicles. In IoV, the trajectory prediction can improve the system performance. In [25], [26], the authors proposed an active load balancing scheme based on the traffic prediction, the balance between the computing loading and the long-term system energy consumption minimization is considered mainly. In [27], the author proposed a non-cooperative game task offloading strategy. However, there existing schemes hadn't considered V2R and V2V offloading jointly and the problem of server switching.

In the highly dynamic network topology and the extremely unstable vehicular communications, if the vehicle selects a target for task offloading, it may cause the task to fail for the reason that the vehicle travels out of the communication range. For example, there is a situation that when the task vehicle offload the task to the RSU or the cooperative vehicle, the task vehicle has already traveled out of the communication range before the task is finished. Especially, with the development of 5G technology, the reduced coverage of RSU and the rapid movement of vehicles will cause frequent RSU handover [28]. Selecting an optimal cooperative vehicle from the surrounding vehicles, the efficiency of V2V task offloading and the task computing completion rate can be guaranteed.

## III. SYSTEM MODEL

Firstly, we present an overview of our proposed task offloading scheme. The vehicles on the road can offload tasks to RSU via V2R communication, or to the surrounding cooperative vehicle via V2V communication links. When the vehicle travels to the overlapped boundary between the two adjacent RSUs, the vehicle has three selections: 1) processing tasks locally, 2) offloading tasks to RSU, and a server switching cost may exist, 3) offloading tasks to the cooperative vehicle. How to select the cooperative vehicle and how to choose the task offloading scheme in VEC are the main problems in this work.

Fig. 1 shows the multi-task offloading scheme between two adjacent RSUs in an urban area. As shown in Fig. 1, the vehicles are divided to mission vehicles and cooperative vehicles. The radius of the two RSUs passing by are $R_1$ and $R_2$. The V2R transmission rate between the vehicle and the RSUs are $r_{R_1}$ and $r_{R_2}$ respectively. Both V2V and V2R communication transmissions use LTE-V technology, which are independent of each other. The sets of mission are $M$ and the cooperative vehicles are $J$. The mission vehicle has multiple tasks, the task $m$ can be expressed as $(L_m, C_m, x_m, T_m^{max})$, $m \in M$, where $L_m$ is the size of the task, $C_m$ is the number of CPU cycles required to process the task, $x_m$ is the location of vehicle when the task $m$ is generated, and $T_m^{max}$ is the maximum processing delay. The vehicle generates multiple tasks randomly. When multiple tasks exist simultaneously, these tasks are sorted according to $T_m^{max}$, and urgent tasks will be processed first.

When the task $m$ is generated under the coverage of RSU1, the tasks can be handled by transferring to RSU1 or cooperative vehicles, and local processing. The result of the tasks offloaded to RSU or vehicles will be returned. Consider the system is operating with a slot-by-slot fashion, we divide the time into multiple time slots, $t \in \{1, 2, \cdots, T\}$. Since the size of calculation result is relatively small, the time and energy loss of the calculation result return are ignored. The channel gains of V2R and V2V, as well as the available computing resources in RSU and cooperative vehicles are time-varying in the considered period, which change with the time slot. Cooperative task offloading is effective in the special time slot, and the offloaded tasks can be completed in the current time slot. the formulation of interference from the nearby vehicles [29]: $I_{V2R} = \sum_{i\in I, i\neq m} P_i^R G_{i,t}^{R_1}$, where $i$ specially refers to the vehicle exclude the current mission vehicle at time slot $t$. The V2R task transmission rate between the mission vehicle and RSU1 at time slot $t$ is denoted as

$$r_{m,t}^{R_1} = b^R \log_2\left(1 + \frac{P^R G_{m,t}^{R_1}}{\sigma^2 + I_{V2R}}\right), \tag{1}$$

where $I_{V2R}$ denotes the interference between multiple V2R transmissions, $\sigma^2$ is Gaussian white noise, $b^R$ represents the bandwidth of the task $m$ transmitted to RSU1, $G_{m,t}^{R_1}$ denotes the channel gain for task transmission to RSU1 at time slot $t$, $P^R$ is transmit power of mission vehicle. Next, we consider three different task offloading schemes.

### A. OFFLOADING TO RSU

When the vehicle chooses offloading tasks to RSU, it may not be completed in the coverage area of the previous RSU (i.e., RSU1). Then, as shown in Fig. 1, it is necessary to transfer part of the transmitted data $L_b$ from RSU1 to RSU2 by the backhaul links. The overhead of backhaul links between RSU1 and RSU2 is expressed as

$$h_b = \frac{c_b L_b p_b}{r_b}, \tag{2}$$

where $r_b$ denotes the transmission rate of the backhaul link between RSU1 and RSU2, and $c_b$ represents the transmission cost. $p_b$ is the transmit power of the backhaul links.

When the vehicle crosses or is about to cross the handover boundary between RSU1 and RSU2, the handover will appear. There are mainly three situations: handover during uplink transmission, computing serving and downlink transmission [30]. Moreover, we believe that handover is mainly in the overlapped communication area of the adjacent RSUs, rather than the communication boundary. At the beginning of each time slot, the vehicle determines the generated tasks whether to switch, the goal is to speed up the task processing. In our work, we focus on the service handover.

Generally, the vehicle generates a series of tasks randomly which can be processed discretely, and the task can decompose into multiple subtasks in the processing stage. Before the vehicle reaches the handover boundary, we set that the task is transmitted to RSU1. The transmission time at time slot $t$ is expressed as

$$T_{m,t}^{R,comm} = \frac{L_m}{r_{m,t}^{R_1}}, \tag{3}$$

where $L_m$ denotes the input size of the task $m$, $r_{m,t}^{R_1}$ represents the transmission rate of task $m$ from mission vehicle to the RSU1 at time slot $t$, as Eq. (1).

If there is no handover during computing serving, task $m$ can be processed in RSU1. Otherwise, it can be processed by RSU2. We set the handover of task $m$ within a certain range before the RSU handover boundary. At the beginning of time slot $t$, the total processed delay of task $m$ transmitted to RSU1 is greater than the total delay of task transmitted to RSU1 and switched to RSU2 for calculation. As

$$T_1 \geq T_2,$$

where $T_1 = \frac{L_m}{r_{m,t}^{R_1}} + \frac{C_m}{f_{m,t}^{R_1}}$, and $T_2 = \frac{L_m}{r_{m,t}^{R_1}} + \frac{L_m}{r_b} + \frac{C_m}{f_{m,t}^{R_2}}$. Then, the server switching exists. The processing frequency of task $m$ at time slot $t$ is shown as

$$f_{m,t}^{R} = \begin{cases} f_{m,t}^{R_1}, & T_1 < T_2 \\ f_{m,t}^{R_2}, & T_1 \geq T_2, \end{cases} \tag{4}$$

where $f_{m,t}^{R_1}, f_{m,t}^{R_2}$ are the allocable CPU frequencies when RSU1 and RSU2 process task $m$ at time slot $t$. The RSU internal calculation time for task $m$ at time slot $t$ can be expressed as

$$T_{m,t}^{R,comp} = \frac{C_m}{f_{m,t}^{R}}, \tag{5}$$

To sum up, the total delay of selecting offloading to the RSU scheme for task $m$ at time slot $t$ is

$$T_{m,t}^{R,total} = \min\{T_1, T_2\}. \tag{6}$$

### B. COMPUTING LOCALLY

To avoid the high cost caused by handover or communication congestion, vehicles can choose to handle the task via itself. In the local task processing scheme of the vehicle, the total delay of task processing is expressed as

$$T_{m,t}^{l,total} = \frac{C_m}{f_{m,t}^{l}}, \tag{7}$$

where $f_{m,t}^{l}$ is the CPU frequency when the vehicle processes task $m$ on its own at time slot $t$.

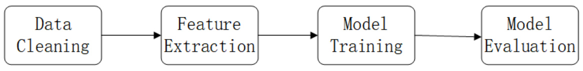


**FIGURE 2. Procedure of trajectory prediction.**

### C. OFFLOADING TO COOPERATIVE VEHICLES

The mission vehicle can offload tasks to the nearby surrounding vehicles. The cooperative vehicles can be selected based on the predicted trajectory. When the available computing resources of RSUs are insufficient, offloading tasks to cooperative vehicles can improve task processing efficiency. Moreover, the server switching cost can be reduced via offloading part/all of the tasks to the cooperative vehicles.

When the tasks offloading to cooperative vehicles via one-hop V2V communication, the offloading scheme includes tasks transmission and calculation. The transmission rate of the vehicle task $m$ from the mission vehicle to the cooperative vehicle $j$ is shown as

$$r_{m,t}^{j} = b^j \log_2\left(1 + \frac{P^j G_{m,t}^{j}}{I_{V2V} + \sigma^2}\right), \tag{8}$$

where $I_{V2V}$ denotes interference, $\sigma^2$ represents Gaussian white noise, $b^j$ is the bandwidth occupied by V2V transmission, $G_{m,t}^{j}$ is the channel gain when the vehicle $j$ transmits the task $m$ to vehicle $j$ at time slot $t$, $P^j$ denotes the transmit power. Therefore, the task transmission delay is expressed as

$$T_{m,t}^{j,comm} = \frac{L_m}{r_{m,t}^{j}}. \tag{9}$$

To sum up, if the task $m$ be offloaded to the cooperative vehicle, the total delay of task processing is expressed as

$$T_{m,t}^{j,total} = T_{m,t}^{j,comm} + T_{m,t}^{j,comp}, \tag{10}$$

where $T_{m,t}^{j,comp}$ denotes the task $m$ calculation delay, and $T_{m,t}^{j,comp} = \frac{C_m}{f_{m,t}^{j}}$, $f_{m,t}^{j}$ denotes the CPU cycle frequency of cooperative vehicle $j$ at time slot $t$.

## IV. OPTIMAL SELECTION OF COOPERATIVE VEHICLE

In this section, we perform the vehicle trajectory prediction, mainly including data processing and model training. Meanwhile, the optimal cooperative vehicle selection of V2V task offloading is proposed based on the trajectory prediction.

### A. DATA PROCESSING

Fig. 2 shows the procedure of trajectory prediction. First, we merge all of the collected data including taxi ID, latitude and longitude, traveling speed, driving direction and recorded time into a dataform, which is sorted by vehicle and time for calculating travel distance and travel speed of each vehicle. Then, the time series analysis problem is transformed into a supervised learning problem. The latitude and longitude of vehicles in the latter moment are combined with the data in the previous moment. Meanwhile, features such as time stamps, travel distance and travel speed at last slot are extracted and merged into the data set. Divide the data set into

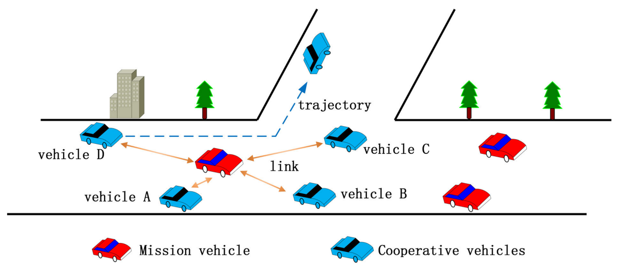


**FIGURE 3.** Problems in cooperative vehicle selection.

training set and testing set, the LightGBM model for training is proposed [31]. The goal of data training is obtain the latitude and longitude of vehicles at the next slot. Moreover, the evaluation metric is the Root Mean Squared Error(RMSE) between the predicted values and the true values, as

$$RMSE = \sqrt{\frac{1}{n}\sum (d_{pred} - d_{actual})^2}, \qquad (11)$$

where $n$ represents the number of measurements, $d_{pred}$ denotes the coordinate prediction of vehicles, and $d_{actual}$ is the actual coordinate of vehicles.

### B. COOPERATIVE VEHICLE SELECTION METHOD

In this section, we analyze the selection of cooperative vehicles and set a decision function to select the optimal cooperative vehicle.

Normally, the dynamic vehicle network topology affects the efficiency of the vehicle communication, which affects the QoS of the mission vehicles. For example, as shown in the Fig. 3, the sudden steering of the cooperative vehicle causes the communication interruption with the mission vehicle. In addition, the allocatable computing capacity and the channel gain of the cooperative vehicle will also affect the QoS. In Fig. 3, the communication distances between the mission vehicle and the cooperative vehicles A, B and C are different, the corresponding channel gains are also different. Meanwhile, the allocatable computing resources of different vehicles are usually different. We summarize these into three factors: allocable computing capacity, average distance traveled, and travel time within the communication range, which we can obtain via the trajectory prediction and data processing.

For we consider the surrounding vehicles belong to the public transportation system, or they belong to the same company of the mission vehicle, we believe that the selected vehicle will help the mission vehicle and obtain the corresponding rewards. According to the trajectory prediction results, the RSU can obtain the average distance between the mission vehicle and the surrounding vehicles $D_j$, and the time duration spent near the mission vehicle $T_j$. We set $D_j = | x_j - x_m |$, where $x_j$ represents the position of cooperative vehicle from dataset. Then, we select the optimal cooperative vehicle based on the following equation, as

$$Ev_j = \underset{j \in J}{argmax}(\alpha_1 C_j - \beta_1 D_j + \omega_1 T_j), \qquad (12)$$

where $\alpha_1, \beta_1, \omega_1$ represents three discount parameters, $\alpha_1 + \beta_1 + \omega_1 = 1$ and $\alpha_1, \beta_1, \omega_1 \in (0, 1)$, $C_j$ denotes the allocatable computing capacity of the surrounding vehicle $j$. It is easy to know that the larger computing capacity of the cooperative vehicle, the closer driving distance and the more staying time duration near to the mission vehicle, the better cooperative vehicle we can choose. After the optimal vehicle is selected, the mission vehicle and the cooperative vehicle $j$ will form a vehicle combination for V2V task offloading.

After data processing and trajectory prediction, the predicted position of vehicles can be obtained, which can be calculated for the communication range $D_{max}$. The mission vehicle searches for cooperative vehicles in communication range $D_{max}$ as the set of the candidate cooperative vehicles $J$. Then, it confirms whether the candidate vehicles will leave the communication range in current time slot. If it does, it will be excluded from the candidate vehicle $J$. Lastly, cooperative vehicle selection is carried out according to Eq.(12). In fact, the selection is based on the quality of communication and calculation. We summarized these into Algorithm 1.

**Algorithm 1** Cooperative Vehicle Selection

```
Require: the time duration traveled near mission vehicle $T_j$;
         the average distance between vehicles $D_j$;
         the allocatable computing capacity $C_j$.
Ensure: the best cooperative vehicle $Ev_j$
  search for cooperative vehicles in communication range
  $D_{max}$ as candidate cooperative vehicles $J$
  for each episode do
    for each vehicle $j$ in candidate cooperative vehicles $J$ do
      if it leaves the communication range with the mission
      vehicles then
        Exclude $j$ from the candidate set $J$
      end if
    end for
    select the cooperative vehicle $Ev_j$ according to Eq.(12)
  end for
```

## V. OPTIMAL TASK OFFLOADING SCHEME

### A. PROBLEM FORMULATION

In this section, we determine the objective function of the task offloading first firstly, which includes two parts: the utility of task transmission and calculation. Meanwhile, to maximize the relative QoS of vehicles with less cost, we define the utility of task offloading as actual offloading income. In other words, the utility of task offloading is defined as the difference between offloading revenue and offloading cost [32].

First, we design the utility of data transmission. Except for the local processing without transmission cost, the transmission cost exists when the task data be transmitted to RSU or the cooperative vehicles. For a certain task $m$ of vehicle, data transmission utility is equal to the difference between network access income expressed as $U^{comm}_{revenue} = a( \lambda^{R}_{m,t} r^{R_1}_{m,t} + \lambda^{V}_{m,t} r^{j}_{m,t})$ and the rental cost of spectrum resources expressed as $U_{cost}^{comm} = \lambda_{m,t}^{R}\delta_m^R b^R + \lambda_{m,t}^{V}\delta_m^j b^j$, Therefore, the utility at time slot $t$ is

$$U_{m,t}^{comm} = U_{revenue}^{comm} - U_{cost}^{comm}, \tag{13}$$

where $a$ denotes unit price per $Mbps$ of network access revenue; $\delta_m^R$, $\delta_m^j$ are the bandwidth leasing cost per $Hz$ when the task is transmitted to the RSU and the cooperative vehicle $j$; $\lambda_{m,t}^{R}, \lambda_{m,t}^{L}, \lambda_{m,t}^{V} \in (0, 1)$ represents the choice of three task offloading options; $b^R$, $b^j$ represents the bandwidth occupied by the task transmitted to RSU and cooperative vehicle $j$ respectively; $r_{m,t}^{R_1}$, $r_{m,t}^{j}$ are the transmission rates from task $m$ to RSU and vehicle $j$ at time slot $t$.

In addition to the task transmission utility, the task calculation utility also exists, which is inevitable for the offloading selection scheme. For a certain task $m$ of mission vehicle, the task calculation utility is equal to the difference between the revenue of task execution expressed as $U_{revenue}^{comp} = b(\lambda_{m,t}^{R}r_{m,t}^{R,comp} + \lambda_{m,t}^{L}r_{m,t}^{local,comp} + \lambda_{m,t}^{V}r_{m,t}^{j,comp})$ and the cost of calculation resources expressed as $U_{cost}^{comp} = \lambda_{m,t}^{R}\eta_i^R\beta_R f_{m,t}^R + \lambda_{m,t}^{L}\eta_m^{local}\beta_l f_{m,t}^l + \lambda_{m,t}^{V}\eta_m^j\beta_j f_{m,t}^j + \lambda_{m,t}^{R}h_b jug(T_1, T_2)$, so the utility at time slot $t$ is

$$U_{m,t}^{comp} = U_{revenue}^{comp} - U_{cost}^{comp}, \tag{14}$$

where the last item is handover cost and we have

$$jug(T_1, T_2) = \begin{cases} 1, & T_1 \geq T_2 \\ 0, & T_1 < T_2, \end{cases} \tag{15}$$

and $b$ denotes the unit revenue of task execution; $\eta_i^R$, $\eta_m^{local}$, $\eta_m^j$ denote the task computing resource costs of the proposed three task transmission schemes. $\beta_R$, $\beta_j$, $\beta_l$ represent the energy consumption generated by each CPU cycle of RSU, the cooperative vehicle and the local vehicle. $\lambda_{m,t}^{R}, \lambda_{m,t}^{L}, \lambda_{m,t}^{V} \in (0, 1)$ represents the selection of three task offloading options. $r_{m,t}^{R,comp}$, $r_{m,t}^{l,comp}$, $r_{m,t}^{j,comp}$ denote the task calculation rates of the three task offloading schemes at time slot $t$ respectively.$f_{m,t}^R, f_{m,t}^l, f_{m,t}^j$ represent the CPU cycle frequency of RSU, task vehicle and cooperative vehicle $j$.

Then, the total utility of all tasks can be expressed as

$$R = \sum_{m \in M}\sum_{t=0}^{T} U_{m,t}^{total}, \tag{16}$$

where

$$U_{m,t}^{total} = U_{m,t}^{comm} + U_{m,t}^{comp}. \tag{17}$$

In this paper,we aim to maximize the long-term utility of vehicular task offloading in the handover coverage area. The joint optimal problem is shown as

$$\max_{\left\{\substack{\lambda_{m,t}^{R}, \lambda_{m,t}^{L}, \lambda_{m,t}^{V} \\ f_{m,t}^R, f_{m,t}^l, f_{m,t}^j, b^R, b^j}\right\}} R \tag{18}$$

subject to:

$$\lambda_{m,t}^{R}, \lambda_{m,t}^{L}, \lambda_{m,t}^{V} = \{0, 1\}, \tag{19}$$

$$\lambda_{m,t}^{R} + \lambda_{m,t}^{L} + \lambda_{m,t}^{V} = 1, \tag{20}$$

$$T_{m,t}^{R,total}, T_{m,t}^{l,total}, T_{m,t}^{j,total} \leq T_m^{\max}, \tag{21}$$

$$f_{m,t}^{R_1} \leq f_m^{R_1,max}, f_{m,t}^{R_2} \leq f_m^{R_2,max}, f_{m,t}^{j} \leq f_m^{j,max}, \tag{22}$$

where constraint (19) and constraint (20) guarantee that each computing task only chooses one offloading scheme such as offloading to RSU, local computing or offloading to cooperative vehicle to execute the computation task at time slot $t$. (21) is the task computation delay constraint, ensuring the total task computation delay of each task processing within the maximum delay limit. (22) is the allocable CPU frequencies constraint, to ensure the allocable CPU frequencies within the maximum CPU frequencies limit.

Obviously, when the task offloading decision variables $\{\lambda_{m,t}^{R}, \lambda_{m,t}^{L}, \lambda_{m,t}^{V}\}$ and the communication computing resource variables $\{f_{m,t}^R, f_{m,t}^l, f_{m,t}^j, b^R, b^j\}$ are determined, the long-term utility of the vehicular task offloading can be obtained. However, $\{\lambda_{m,t}^{R}, \lambda_{m,t}^{L}, \lambda_{m,t}^{V}\}$ are the binary variables, which makes this problem not convex and it difficult to be solved directly. In addition, when the vehicle performs task offloading, the communication environment of VEC, the moving trajectory of the cooperative vehicles and the available computing resources of RSU and cooperative vehicles are all time-varying. In the dynamical environment, when the task offloading decision is set as the action selection, it conforms to the background of reinforcement learning. Thus, DRL method is used to solve this problem.

## B. DRL-BASED TASK OFFLOADING SCHEME

At the handover area, the vehicle has multiple tasks to be processed, and the maximum completion time limits $T_m^{max}$. According to $T_m^{max}$, we sort the tasks from low to high and process the lower one firstly. In this section, we use DQN to solve the task offloading problem in the handover area in IoV. DQN is a very common use of DRL to simulate the decision-making process [32], [33]. We define the task processing as a Markov decision process. Meanwhile, we define the system state space, decision action space and reward function based on the task offloading long-term utility function proposed previously.

### 1) STATE SPACE

The task offloading utility is closely related to the task communication and calculation processes. the channel gains $G_{m,t}^j$, $G_{m,t}^R$ are related to the task transmission, the CPU frequency of the RSU $f_{m,t}^R$, and the cooperative vehicle $j$, $f_{m,t}^j$ are related to the task calculation, $t \in [0, 1, \cdots, T-1]$. The system state is expressed as

$$s_t = [G_{m,t}^j, G_{m,t}^R, f_{m,t}^R, f_{m,t}^l, f_{m,t}^j]. \tag{23}$$

### 2) ACTION SPACE

In the DQN model, the decision agent needs to determine the offloading strategy of the computing tasks, which can be offloaded to the RSU, be processed locally or be offloaded to the cooperative vehicle. Therefore, we define the offloading strategy as a binary variable corresponding to the offloading scheme,where $\lambda_{m,t}^{R}, \lambda_{m,t}^{L}, \lambda_{m,t}^{V}$ represent the selection of the three offloading schemes. Hence, the action space of task $m$ is expressed as

$$a_t = [\lambda_{m,t}^{R}, \lambda_{m,t}^{L}, \lambda_{m,t}^{V}]. \tag{24}$$

### 3) REWARD FUNCTION

We have determined the objective function for vehicular computing task offloading in the previous system model to represent the long-term utility of the system. Then, the objective function is used as the reward function in DQN model. We assume although the mission vehicle have multiple computing tasks, all of the tasks in a time slot can be transmitted and processed. The reward of task $m$ at time slot $t$ is given as

$$\begin{aligned} r_t &= U_{m,t}^{total} = U_{m,t}^{comm} + U_{m,t}^{comp} \\ &= a(\lambda_{m,t}^{R} r_{m,t}^{R_1} + \lambda_{m,t}^{V} r_{m,t}^{j}) - \lambda_{m,t}^{R}\delta_m^R b^R - \lambda_{m,t}^{V}\delta_m^j b^j \\ &\quad + b(\lambda_{m,t}^{R} r_{m,t}^{R,comp} + \lambda_{m,t}^{L} r_{m,t}^{local,comp} + \lambda_{m,t}^{V} r_{m,t}^{j,comp}) \\ &\quad - \lambda_{m,t}^{R}\eta_i^R \beta_R f_{m,t}^{R} - \lambda_{m,t}^{L}\eta_m^{local}\beta_l f_{m,t}^{l} - \lambda_{m,t}^{V}\eta_m^j \beta_j f_{m,t}^{j} \\ &\quad - \lambda_{m,t}^{R} h_b jug(T_1, T_2). \end{aligned} \tag{25}$$

The optimal action can be obtained when the reward is maximum, as

$$a_t^* = \underset{a_t}{argmax}\, r_t(s_t, a_t). \tag{26}$$

We can obtain immediate reward $r_t$ via selecting the action $a_t^*$ in time slot $t$. The objective of our DQN model is to maximize the cumulative reward, as

$$\overline{R} = \max \sum_{m \in M} \sum_{t=0}^{T} r_t. \tag{27}$$

When the agent is in the state $s_t$, take action $a_t$, it will get reward $r$, the next state $s_{t+1}$ can be expressed as

$$s_t \xrightarrow{a_t} r_t, s_{t+1}. \tag{28}$$

In this case, we define a Q function $Q(s_t, a_t)$ to evaluate the reward at time slot $t$, is shown as

$$\begin{aligned} Q_{new}(s_t, a_t) = Q_{now}(s_t, a_t) &+ \alpha[r_t + \gamma \max_{a'} Q'(s', a') \\ &- Q_{now}(s_t, a_t)], \end{aligned} \tag{29}$$

where $Q_{new}(s_t, a_t)$ denotes the new Q function value, $Q_{now}(s_t, a_t)$ is the current one, $\alpha$ represents the learning rate, $r_t$ denotes reward at time slot $t$, $\gamma$ represents discount factor and $Q'(s', a')$ is the Q value under a given new state and action.

In the iteration of each time slot, the Q function is updated as

$$\begin{aligned} Q(s_t, a_t) \leftarrow Q(s_t, a_t) &+ \alpha[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \\ &- Q(s_t, a_t)]. \end{aligned} \tag{30}$$

Due to the large state and action space, employing Q learning directly to solve the optimization problem of this article is not effective. Via using a neural network to simulate the Q function, which is approximated as: $Q(s_t, a_t) \approx Q(s_t, a_t; \theta)$, where $\theta$ represents the parameters of the predictive neural network, DQN can be used to solve the complex problem. In order to minimize the difference between $Q(s_t, a_t)$ and $Q(s_t, a_t; \theta)$ for training, we formulate the mean square error (MSE) loss function as

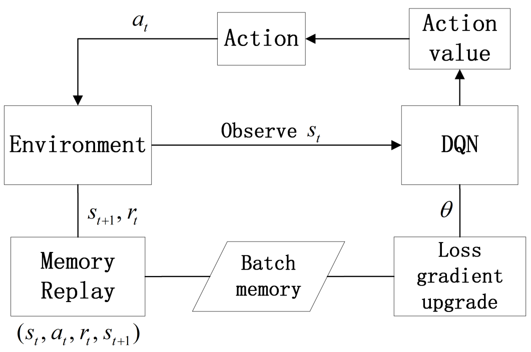


**FIGURE 4.** **DQN model of data offloading problem.**

$$Loss(\theta_t) = \frac{1}{m}\sum_{j=1}^{m}(y_t - Q(s_t, a_t; \theta_t))^2, \tag{31}$$

where $y_t$ represents the Q values of target network, which is given as

$$y_t = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_t^-). \tag{32}$$

Then we use the gradient update method to minimize the loss function for updating the evaluated Q network parameters. Specially, the parameter $\theta$ is updated in the direction of the minimized loss function, as

$$\theta_{t+1} = \theta_t + \alpha[y_t - Q(s_t, a_t; \theta_t)]\nabla_{\theta_t} Q(s_t, a_t; \theta_t). \tag{33}$$

The proposed DQN model includes a behaviour Q network and a target Q network, where the set of parameters of the them are $\theta$ and $\theta^-$, respectively. Employing the experience replay mechanism to update network parameters, we replace a certain size of experience replay memory $C$, in which multiple experience samples containing actions, states and rewards can be stored [34]. The specific structure of DQN model is shown as Fig. 4.

In order to avoid reaching the local optimal point, we adopt the $\varepsilon$-greedy policy, which is to choose random actions with probability $\varepsilon$, otherwise, the optimal action are selected from the experience playback pool. After performing the selection action, the execution reward and the state of the next moment can be obtained. Therefore, the state transition $(s_t, a_t, r_t, s_{t+1})$ is obtained for storing in the experience replay memory. In the neural network learning phase of each time slot, the decision agent in DQN model randomly selects a small sample of the experience replay memory to update network parameters $\theta$, which can suppress the time correlation of the data. Then gradient descent is performed to minimize the square error to update the estimated Q network parameters. In summary, the DQN-based TO scheme in handover area in VEC algorithm is shown as Algorithm 2.

**Algorithm 2** DQN-Based to Scheme in Handover Area

```
Require: random state space $s_t$, action space $a_t$;
         parameters $\theta$ of behavior Q network and
         the parameters $\theta^-$ of target network;
         the capacity of the experience replay memory $C$.
Ensure: maximum long-term utility of offloading system $\overline{R}$.

1:  for each episode do
2:      Obtain the initial state $s_0$;
3:      for $t = [0, 1, 2, \cdots, T-1]$ do
4:          Randomly select action $a_t$ with probability $\varepsilon$;
5:          Otherwise choose action according to
            $a_t = arg \max_{a_t} r_t(s_t, a_t; \theta_t)$;
6:          Execute action $a_t$, derive the next state $s_{t+1}$ and
            obtain offloading utility $r_t$;
7:          Store $(s_t, a_t, r_t, s_{t+1})$ in the experience replay memory;
8:          Sample random mini-batch of transitions from
            experience replay memory and calculate the loss
            function Eq. (31)
9:          Update the weight of the DQN $\theta$ by calculating the
            gradient of loss function by Eq. (33);
10:         Update $\theta^-$ according to $\theta$ lately;
11:     end for
12: end for
```

## VI. EXPERIMENTAL SIMULATION

In this part, we conduct simulation to verify the impact of DQN-based TO scheme on the performance of our proposed task offloading system. Firstly, we introduce the simulation scenarios and parameter settings. Then, we analyze the simulation results.

### A. SIMULATION SETTING

We collect traveling data from the Internet. The trajectory data comes from the traffic line access time prediction of the DC competition on websites: https://js.dclab.run/v2/cmptDetail.html?id=318. It is a public data set about collected data of taxi history trajectory. The taxi data file includes the recorded data such as taxi ID, latitude and longitude, traveling speed, driving direction and recorded time. We consider that vehicles traveling at a real traveling speed passes through the two adjacent RSUs' handover area, and there are some surrounding vehicles traveling nearby. The communication range of the RSU is 800 m, and the intersection between the two RSUs is 300 m, which is assumed to be handover area. The vehicle has multiple tasks to be processed, which can be sorted according to the delay limit and processed in order. In each iteration, the vehicle travels from the start point to the end point crosses the handover area. At the beginning, we set $x_m = 0$ represents the initial task generation position of the vehicle, and set the traveling speed according to the processing data. We present small part of data processing result in Table 1. At each time slot, the state

**TABLE 1.** Trajectory data after processing.

| taxiID | lng | lat | direction | velocity |
|---|---|---|---|---|
| $BT9**bfce81$ | 120.187 | 35.9627 | 135 | 0.000000 |
| $BT9**bfce81$ | 120.187 | 35.9625 | 90 | 37.361495 |
| $BT9**bfce81$ | 120.187 | 35.9597 | 90 | 32.009052 |
| $BT9**bfce81$ | 120.187 | 35.9588 | 90 | 37.361495 |
| $BT9**bfce81$ | 120.187 | 35.9560 | 88 | 32.243327 |
| $BT9**bfce81$ | 120.187 | 35.9542 | 116 | 30.568569 |
| $BT9**bfce81$ | 120.185 | 35.9533 | 127 | 10.532450 |
| $BT9**bfce81$ | 120.184 | 35.9531 | 98 | 23.926379 |

space is different, the mission vehicle processes the generated task by selected the optimal offloading scheme according to the state space, which is regarded as a training step. Until the vehicle exits the handover area, the iteration ends and enters the next iteration.

In the communication environment, the channel gains of V2R and V2V are generated according to the data set after data processing. For task $m$, the size $L_m$ is generated in $[1, 2]MB$ and each $1Mb$ correspond to $C_m = 500$. Meanwhile, all the CPU frequency related to the task load. When the cooperative vehicles or RSUs perform task executing, they assigns all of their computing capacity. The simulation parameters are summarized in Table 2.

We are using the environment of tensorflow 1.13.1 version. The behavior Q network and target Q network of our DRL model both set up a two-layer network. The size of the experience pool in RL is 2000 and each time take out 32 samples. The reward decay factor set to be 0.9, the greedy probability is 0.9.

### B. SIMULATION RESULTS

#### 1) TRAJECTORY PREDICTION

We bring the dataset into the model for training. Since the dataset brought into GBM must be a 1-D numpy array, we bring the two coordinates of longitude and latitude as a 1-D numpy array into the GBM model for training separately. The training process of the two GBM models is shown as the Fig. 5. The $valid_0$ represents the RMSE value of the validation set and the *training* shows the RMSE value of the training set.

It can be seen that there is no significant difference in the training error for the longitude and latitude as the input data of the LightGBM model. The reason is that both of them use the same extracted features for the model training, the square root of RMSE of current longitude and latitude can replace the current position, which is given as

$$RMSE_{pos} = \sqrt{RMSE_{lat}^2 + RMSE_{lng}^2}. \tag{34}$$

Fig. 6 is the training process of LightGBM model, which shows the comparison of RMSE between the training set and the validation set. It is easy to find that the RMSE of validation set is larger than the training one, but they are similar. We can generally assume that the trajectory prediction is accurate.

**TABLE 2. Default parameter setup.**

| Parameter | Definition | Value |
|---|---|---|
| $\sigma^2$ | Background noise | $-19dbm$ |
| $b^R, b^j$ | Spectrum bandwidth of V2R, V2V | $10, 6MHz$ |
| $P^R, P^j$ | Transmit power of V2R, V2V | $20, 20dBm$ |
| $T_m^{max}$ | Maximum delay of task execution | $0.1s$ |
| $r_b$ | Transmission rate of backhaul link | $5Mb$ |
| $p_b$ | Transmission power of backhaul link | $1W$ |
| $c_b$ | the unit cost of backhaul link | $4J$ |
| $a$ | Revenue of task transmission access | $2units/Mbps$ |
| $b$ | Revenue of task execution | $2units/Mbps$ |
| $\delta_i^R$ | V2R bandwidth lease fee | $7.0units/MHz$ |
| $\delta_i^j$ | V2V bandwidth lease fee | $4.7units/MHz$ |
| $\beta_R$ | Energy consumption of RSU | $40W/GHz$ |
| $\beta_i$ | Energy consumption of local | $70W/GHz$ |
| $\beta_j$ | Energy consumption of vehicle | $70W/GHz$ |
| $\eta_{i,R}$ | Expense of RSU calculation | $0.10units/W$ |
| $\eta_{i,local}$ | Expense of local calculation | $0.04units/W$ |
| $\eta_{i,j}$ | Expense of vehicle calculation | $0.06units/W$ |
| $f_{m,max}^{R_1}, f_{m,max}^{R_2}$ | Max Calculation frequency of RSUs | $40, 60GHz$ |
| $f_{m,max}^j$ | Max calculation frequency of vehicles | $30GHz$ |

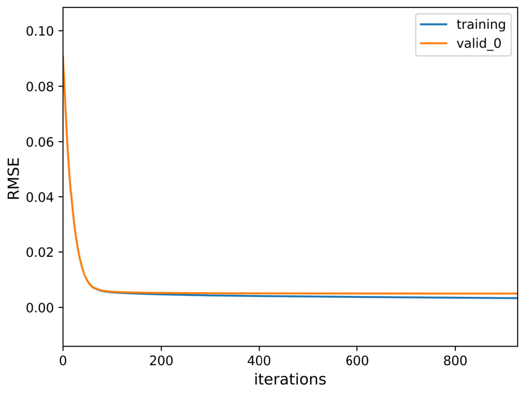
Fig. 7 shows the rewards of DQN-based TO scheme with different V2V cooperative vehicles of trajectory prediction. Selecting different V2V cooperative vehicles for tasks offloading which have different average distance between mission vehicle and computing capabilities. Similar as Fig. 3, we ensure that the best vehicle is vehicle A via the trajectory prediction. Besides, we choose other three vehicles from data set for simulation to make a contrast. The vehicle B has shorter average distance between mission vehicle, while vehicle C has less computing capabilities. And we choose vehicle D which will drive away from the mission vehicle. From the Fig. 7, we can find that the DQN-based TO Scheme with trajectory prediction get the best reward. Obviously, when the cooperative vehicle choose vehicle B, vehicle C or vehicle D, it will not reach the maximum profit. The trajectory prediction is related to the cooperative vehicle selection for V2V communication.

### 2) DQN-BASED TO SCHEME

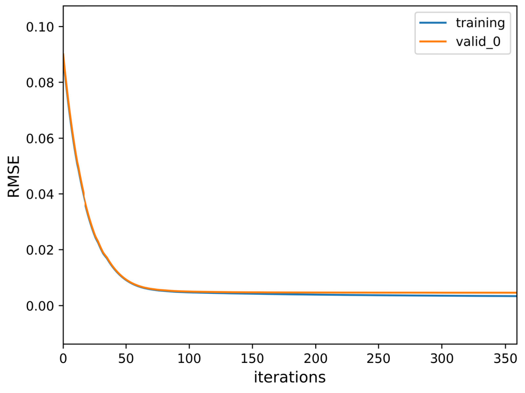
In order to conduct the performance of the proposed DQN-based TO scheme, we simulate five baselines schemes as follows:

- **Q-learning TO Scheme**: Q-learing is used for the offloading scheme.
- **Sarsa TO Scheme**: Sarsa is used for the offloading scheme.
- **Offloading to RSU (OtRSU)**: the tasks only be offloaded to RSU;
- **Computing locally (CLocal)**: the tasks only be processed locally;
- **Offloading to cooperative vehicles (OtCVs)**: the tasks only be offloaded to cooperative vehicles.

Fig. 8 shows the relationship between the rewards of different offloading schemes and the number of iterations. We set the learning rate of all offloading schemes as 0.01, the total number of running iterations is 1000, and all programs start learning once in each iteration after 25 iterations. The DQN-based TO scheme reaches its optimal value around



(a) Latitude target for model training.



(b) Longitude target for model training.

**FIGURE 5. Longitude and latitude target for model training.**

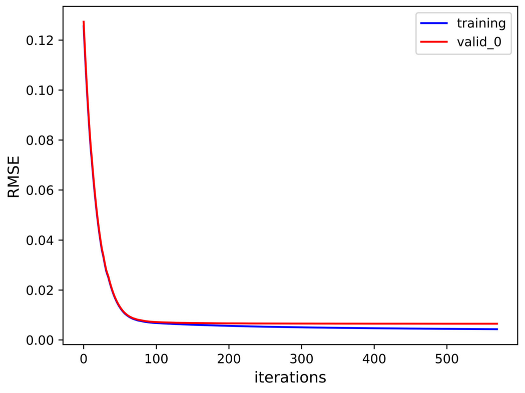


**FIGURE 6. The multi-task offloading scheme model.**

220 iterations. Besides, Q-learning TO Scheme and Sarsa TO Scheme reaches their optimal values around 100 iterations. We can find that the rewards obtained by the DQN-based TO scheme are much higher than other benchmark schemes. Besides, Q-learning TO Scheme and Sarsa TO Scheme achieve similar optimal value. This is because DQN-based TO scheme uses the most effective offloading scheme in each time slot. Meanwhile, DQN uses neural network to replace Q table and update the weights of the network by training a small batch data from memory replay.

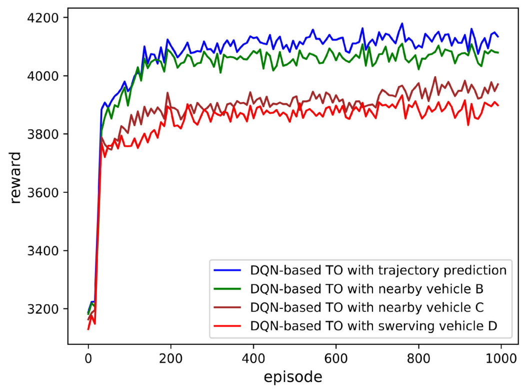


**FIGURE 7.** DQN-based TO scheme rewards with different V2V cooperative vehicles of trajectory prediction.

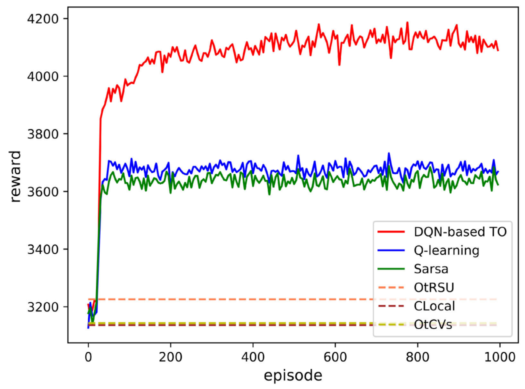


**FIGURE 8.** Rewards of different offloading schemes.

Obviously, other three schemes: OtRSU, CLocal and OtCVs cannot achieve the best system performance for the current time slot.

Fig. 9 shows the execution time of different offloading schemes. The simulation parameters are the same as Fig. 8. From the Fig. 9, we can observe that the DQN-based TO Scheme has a similar execution time with Sarsa offloading scheme, Q-learning offloading scheme has a slightly larger execution time. Besides, OtRSU, CLocal and OtCVs do not need to train the action network which have lower execution time.

Fig. 10 shows the rewards of different offloading schemes and transmit power. We set transmit power $P = P^R = P^j$. From the Fig. 10, we can find that most baseline schemes all increase with the increase of the transmission power $P$. This is because the DQN-based TO scheme, Q-learning TO Scheme, Sarsa TO Scheme, OtRSU and OtCVs are all related to the transmission power $P$, which affects the task transmission rate directly. Since the local processing task does not require task transmission, it does not affect the CLocal scheme. Meanwhile, as the transmission power increasing, the reward of Q-learning TO Scheme and Sarsa TO Scheme begin to be lower than OtRSU and CLocal. In addition, after a certain number of iterations of training, the reward of the DQN-based TO scheme after convergence is higher than that of the baseline schemes. The performance of DQN-based TO scheme is the best among the proposed schemes.

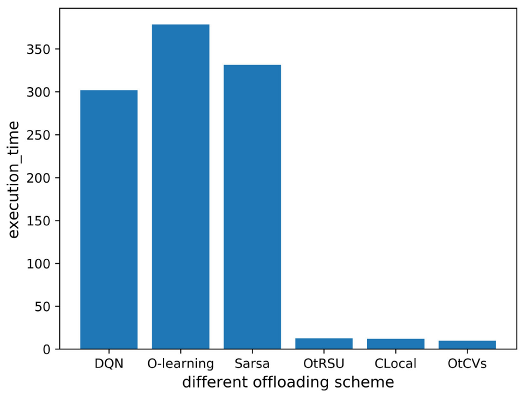


**FIGURE 9.** Execution time of different offloading schemes.

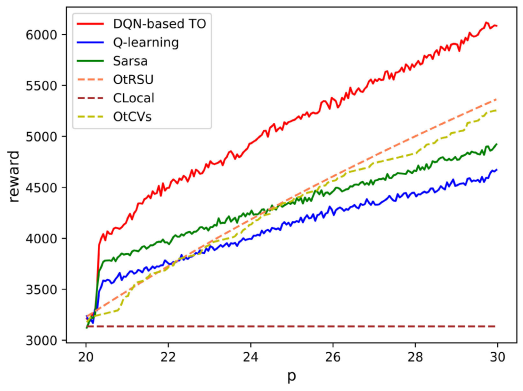


**FIGURE 10.** Rewards of different offloading schemes under the transmit powers.

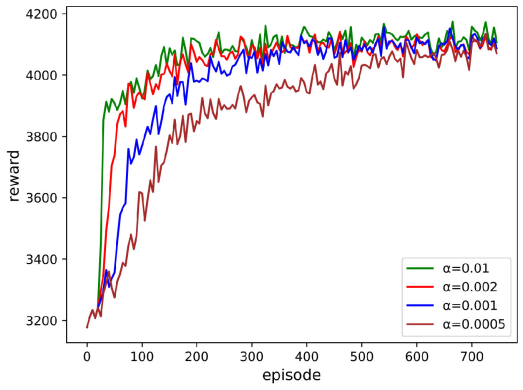


**FIGURE 11.** Rewards of DQN-based TO scheme with different learning rates.

Fig. 11 shows the relationship between the rewards of DQN-based TO Scheme with different learning rates and the number of iterations. In Fig. 11, the learning rates $\alpha$ of DQN are set as 0.01, 0.002, 0.001 and 0.0005. It can be seen that the higher the learning rate, the faster the convergence speed, and in the end it can converge to the same optimal value, which is in line with expectations. Properly adjusting

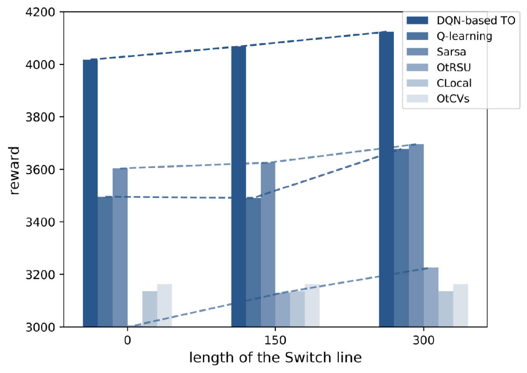


**FIGURE 12.** **DQN-based TO scheme rewards with different handover area.**

the learning rate can improve the convergence speed of DQN directly.

Fig. 12 shows the rewards for different task offloading schemes with different handover area. We set the length of the switching zone be $[0, 150, 300]m$. The total number of running iterations is 1000. It can be seen that as the length of the handover area becomes longer, the reward of OtRSU, Q-learning TO Scheme, Sarsa TO Scheme and the DQN-based TO Scheme increase. This is because we mainly consider the service switching cost of the RSU which leads to an increasing in reward of the above schemes. Besides, DQN-based TO Scheme, Q-learning TO Scheme, Sarsa TO Scheme selects the optimal scheme at each time slot, and update their action values in their own way at each iteration. Therefore, we can find that the timely handover can help the system obtain greater utility.

## VII. CONCLUSION

In this paper, we propose a DQN-based TO scheme in handover area of VEC based on trajectory prediction, which focuses on the serving handover between RSUs and the selecting of cooperative vehicles. For time-varying environment of IoV, we present a cooperative vehicle selection method based on trajectory prediction and then propose an optimal task offloading scheme based on DRL, which vehicles can cooperate with RSUs or moving vehicles for task processing. In order to maximize the long-term offloading utility, we use DQN to simulate the comprehensive offloading scheme. The simulation results show that, considering the selection of cooperative vehicles in V2V communication and the servers switching of RSUs, our proposed task offloading scheme has great advantages in improving the utility of vehicle computing task offloading.

## REFERENCES

[1] F. T. Monteiro, W. S. Costa, J. L. C. Neves, D. M. I. Silva, H. R. O. Rocha, E. O. T. Salles, and J. A. L. Silva, "Experimental evaluation of pulse shaping based 5G multicarrier modulation formats in visible light communication systems," *Opt. Commun.*, vol. 457, Feb. 2020, Art. no. 124693.

[2] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 976–986, Feb. 2019.

[3] S. Bi and Y. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[4] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 242–253, Jan. 2021.

[5] R. Bi, Q. Liu, J. Ren, and G. Tan, "Utility aware offloading for mobile-edge computing," *Tsinghua Sci. Technol.*, vol. 26, no. 2, pp. 239–250, Apr. 2021.

[6] W. Feng, J. Xu, and Z. Ding, "Multi-antenna NOMA for computation offloading in multiuser mobile edge computing systems," *IEEE Trans. Commun*, vol. 67, no. 3, pp. 2450–2463, Mar. 2019.

[7] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.

[8] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.

[9] H. Zhang, Z. Wang, and K. Liu, "V2X offloading and resource allocation in SDN-assisted MEC-based vehicular networks," *China Commun.*, vol. 17, no. 5, pp. 266–283, May 2020.

[10] S.-R. Yang, Y.-J. Su, Y.-Y. Chang, and H.-N. Hung, "Short-term traffic prediction for edge computing-enhanced autonomous and connected cars," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3140–3153, Apr. 2019.

[11] J. Du, C. Jiang, J. Wang, Y. Ren, and M. Debbah, "Machine learning for 6G wireless networks: Carrying forward enhanced bandwidth, massive access, and ultrareliable/low-latency service," *IEEE Veh. Technol. Mag.*, vol. 15, no. 4, pp. 122–134, Dec. 2020.

[12] J. Tang, R. Yu, S. Liu, and J. Gaudiot, "A container based edge offloading framework for autonomous driving," *IEEE Access*, vol. 8, pp. 33713–33726, 2020.

[13] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-based task offloading for vehicular cloud computing systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.

[14] H. Ye and G. Y. Li, "Deep reinforcement learning for resource allocation in V2V communications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[15] L. Huang, S. Bi, and Y.-J.-A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[16] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8050–8062, Aug. 2019.

[17] Y. Xie, Z. Xu, Y. Zhong, J. Xu, S. Gong, and Y. Wang, "Backscatter-assisted computation offloading for energy harvesting IoT devices via policy-based deep reinforcement learning," in *Proc. IEEE/CIC Int. Conf. Commun. Workshops China (ICCC)*, Aug. 2019, pp. 65–70.

[18] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[19] X. Zhao, K. Yang, Q. Chen, D. Peng, H. Jiang, X. Xu, and X. Shuang, "Deep learning based mobile data offloading in mobile edge computing systems," *Future Gener. Comput. Syst.*, vol. 99, pp. 346–355, Oct. 2019.

[20] J. Du, C. Jiang, H. Zhang, Y. Ren, and M. Guizani, "Auction design and analysis for SDN-based traffic offloading in hybrid satellite-terrestrial networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2202–2217, Oct. 2018.

[21] J. Du, E. Gelenbe, C. Jiang, H. Zhang, and Y. Ren, "Contract design for traffic offloading and resource allocation in heterogeneous ultra-dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2457–2467, Nov. 2017.

[22] R. Atallah, C. Assi, and M. Khabbaz, "Deep reinforcement learning-based scheduling for roadside communication networks," in *Proc. 15th Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, May 2017, pp. 1–8.

[23] Y. Wu, J. Wu, L. Chen, J. Yan, and Y. Luo, "Efficient task scheduling for servers with dynamic states in vehicular edge computing," *Comput. Commun.*, vol. 150, pp. 245–253, Jan. 2020.

[24] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, and X. Shen, "Toward efficient content delivery for automated driving services: An edge computing solution," *IEEE Netw.*, vol. 32, no. 1, pp. 80–86, Jan./Feb. 2018.

[25] J. Li, G. Luo, N. Cheng, Q. Yuan, Z. Wu, S. Gao, and Z. Liu, "An end-to-end load balancer based on deep learning for vehicular network traffic control," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 953–966, Feb. 2019.

[26] X. Gao, X. Huang, S. Bian, Z. Shao, and Y. Yang, "PORA: Predictive offloading and resource allocation in dynamic fog computing systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 72–87, Jan. 2020.

[27] Z. Xiao, X. Dai, H. Jiang, D. Wang, H. Chen, L. Yang, and F. Zeng, "Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2038–2052, Mar. 2020.

[28] S. Wan, R. Gu, T. Umer, K. Salah, and X. Xu, "Toward offloading Internet of Vehicles applications in 5G networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4151–4159, Jul. 2020.

[29] M. Dai, Z. Su, Q. Xu, and N. Zhang, "Vehicle assisted computing offloading for unmanned aerial vehicles in smart city," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1932–1944, Mar. 2021.

[30] V. Huy Hoang, T. M. Ho, and L. B. Le, "Mobility-aware computation offloading in MEC-based vehicular wireless networks," *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 466–469, Feb. 2020.

[31] X. Hu, H. Chen, and R. Zhang, "Short paper: Credit card fraud detection using LightGBM with asymmetric error control," in *Proc. 2nd Int. Conf. Artif. Intell. Ind. (AI4I)*, Sep. 2019, pp. 91–94.

[32] Z. Ning, P. Dong, X. Wang, J. J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, p. 60, 2019.

[33] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Aug. 2019.

[34] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

**BAIQUAN LV** received the B.E. degree from Guangdong University of Technology (GDUT), Guangzhou, China, in 2019, where he is currently pursuing the M.S. degree with the School of Automation. His research interests include wireless communication networks, cooperative communications, and intelligent edge computing.

**CHAO YANG** received the Ph.D. degree in signal and information processing from the South China University of Technology, Guangzhou, China, in 2013. From 2014 to 2016, he was a Research Associate with the Department of Computing, The Hong Kong Polytechnic University. He is currently with the School of Automation, Guangdong University of Technology. His research interests include VANETs, smart grid, and edge computing.

**XIN CHEN** received the Ph.D. degree in bioinformatics from Harbin Medical University, Harbin, China, in 2012. After that, she was a Postdoctoral Fellow with the Faculty of Health Sciences, University of Macau. Since 2016, she has been working with the Institute of Intelligent Information Processing, Guangdong University of Technology (GDUT). Her research interests include computational biology, wireless big data analysis, and convex optimization.

**ZHIHUA YAO** received the B.E. degree from Guangdong University of Technology (GDUT), Guangzhou, China, in 2019, where he is currently pursuing the M.S. degree with the School of Automation. His research interests include wireless communication networks, cooperative communications, and intelligent edge computing.

**JUNJIE YANG** received the B.E. degree from the Southwest University of Science and Technology, Mianyang, China, in 2008, and the Ph.D. degree from the School of Automation, Guangdong University of Technology, Guangzhou, China, in 2017. From 2016 to 2017 and from 2018 to 2019, he was a Research Fellow with the Department of Centre for Research in Mathematics, Western Sydney University, Australia. He is currently a Lecturer with the School of Automation, Guangdong University of Technology. His research interests include wireless networks, edge computing, and machine learning.