Collaborative Data Scheduling for Vehicular Edge Computing via Deep Reinforcement Learning

Quyuan Luo, Changle Li, *Senior Member, IEEE*, Tom H. Luan, *Senior Member, IEEE*, and Weisong Shi, *Fellow, IEEE*

***Abstract*—With the development of autonomous driving, the surging demand for data communications as well as computation offloading from connected and automated vehicles can be expected in the foreseeable future. With the limited capacity of both communication and computing, how to efficiently schedule the usage of resources in the network toward best utilization represents a fundamental research issue. In this article, we address the issue by jointly considering the communication and computation resources for data scheduling. Specifically, we investigate on the vehicular edge computing (VEC) in which edge computing-enabled roadside unit (RSU) is deployed along the road to provide data bandwidth and computation offloading to vehicles. In addition, vehicles can collaborate among each other with data relays and collaborative computing via vehicle-to-vehicle (V2V) communications. A unified framework with communication, computation, caching, and collaborative computing is then formulated, and a collaborative data scheduling scheme to minimize the system-wide data processing cost with ensured delay constraints of applications is developed. To derive the optimal strategy for data scheduling, we further model the data scheduling as a deep reinforcement learning problem which is solved by an enhanced deep *Q*-network (DQN) algorithm with a separate target *Q*-network. Using extensive simulations, we validate the effectiveness of the proposal.**

***Index Terms*—Data scheduling, deep *Q*-network (DQN), deep reinforcement learning (DRL), vehicular edge computing (VEC).**

I. INTRODUCTION

IN RECENT years, there are increasing interest in connected and automated vehicles, which integrate information and communication technologies and play a crucial role toward a safer and more intelligent transportation system [1]. However, the increasing number of connected and automated vehicles and their resource-hungry applications pose great challenges to the limited capability of vehicles in terms of data computing capacity for providing real time and reliable vehicular services [2]. The paradigm of vehicular edge computing (VEC) has been come up accordingly to strengthen the service capability through moving computation nodes to the proximity of vehicles [3].

In a VEC network, edge computational nodes (ECNs) can be deployed in cell towers, roadside units (RSUs), and within connected and automated vehicles. Although offloading data to ECNs can significantly improve the delay performance, it also increases the burden of radio spectrum resource when data are transmitted through radio access networks [4], [5]. The sharp increase in the demand for communication during data offloading imposes great challenges for the existing communication resources of the VEC network [6]. Accordingly, it is vital to jointly consider the communication and computation resources for data scheduling to alleviate the situation.

Some existing works have focused on the joint allocation of communication and computation resources [7]–[10]. In these work, the data scheduling is mostly formulated as a resource scheduling problem through either minimizing the total delay or cost, or maximizing the system utility. Marvelous solutions are proposed to solve these optimization problems. However, the dynamic and changing VEC environment make the resource allocation a nonconvex optimization problem with the complicated objection function and constraints, which are much complicated and difficult to solve [11]. Good at abstracting many factors that affect the VEC environment to a mapping problem and learning optimal resource allocation strategy from the environment, deep reinforcement learning (DRL) is becoming a research hotspot to solve the resource allocation problem [12], [13]. Tan and Hu [14] proposed a DRL-based joint communication, caching, and computing allocation scheme in vehicle networks. He *et al.* [15] proposed a dueling-DQN (DDQN)-based approach to solve their formulated joint optimization problem of networking, caching, and computing resources for connected vehicles. More recently, Zhang *et al.* [16] proposed a deep *Q*-network (DQN)-empowered task offloading for MEC in urban informatics.

However, most of the existing works consider offloading data to RSU or cell towers, ignoring the idle computing capacity of collaborative vehicles. Since vehicles themselves can be regarded as ECNs, they can assist data processing for other vehicles if they have some idle computing resources, which will greatly reduce the data processing cost and burden of

Manuscript received December 15, 2019; revised February 26, 2020; accepted March 13, 2020. Date of publication March 26, 2020; date of current version October 9, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant U1801266, in part by the National Key Research and Development Program of China under Grant 2019YFB1600100, in part by the Key Research and Development Program of Shaanxi under Grant 2018ZDXM-GY-038 and Grant 2018ZDCXL-GY-04-02, in part by the Youth Innovation Team of Shaanxi Universities, in part by the Science and Technology Projects of Xi'an, China, under Grant 201809170CX11JC12, and in part by the China Scholarship Council. *(Corresponding author: Changle Li.)*

Quyuan Luo and Changle Li are with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China (e-mail: qyluo@stu.xidian.edu.cn; clli@mail.xidian.edu.cn).

Tom H. Luan is with the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: tom.luan@xidian.edu.cn).

Weisong Shi is with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA (e-mail: weisong@wayne.edu).

Digital Object Identifier 10.1109/JIOT.2020.2983660

RSUs or cell towers. In light of the existing works, in this article, we make a further step in designing, analyzing, and optimizing data scheduling through jointly consider communication, computation, caching, and collaborative computing a unified framework. Inspired by the DRL, we further propose a collaborative data scheduling scheme for the VEC network based on DQN, to minimize data processing cost while ensuring delay constraints. Specifically, the contributions of this article can be summarized as follows.

1) *Model:* We model the data scheduling model in a unified VEC network, where data can be processed locally, offloaded to RSUs, migrated to collaborative vehicles, or kept in caching queues. Considering the remaining lifetime and caching state of data, we establish a multiqueue model for data caching on both the vehicle side and RSU side.
2) *Algorithm Design:* To derive the optimal data scheduling strategy, we first formulate an optimization problem to minimize the system-wide data processing cost while ensuring delay constraints. Then, the data scheduling is modeled as a DRL problem to reflect the interaction between data scheduling and cost, by jointly considering communication resource, caching states of data, computing resource, delay requirements of data, and the mobility of vehicles.
3) *Validation:* Based on the real-world vehicular trace, the proposed scheme is evaluated by extensive simulations. The simulation results validate the performance of our proposal and show that our proposal efficiently reduces data processing cost and helps data be processed under delay constraints.

The remainder of this article is organized as follows. The related work is presented in Section II. In Section III, we depict the system model and establish a multiqueue model for data caching on dual sides. The data scheduling problem is formulated in Section IV. The DQN-based scheduling scheme is introduced in Section V. In Section VI, extensive simulation results are discussed. The conclusion is drawn in Section VII.

II. Related Work

In this section, we survey the existing literature on joint resource allocation of communication and computation for data scheduling in the VEC system.

Du *et al.* [19] proposed an online DDORV algorithm, which utilizes the Lyapunov optimization theory to minimize the averaged cost of MEC-enabled roadside units (MRSUs) and vehicular terminal. Through derivation and comparing the values of local processing cost and task offloading cost, the optimization problem on the vehicular terminal side is solved. For the optimization issue on the MEC server side, the Lagrangian dual decomposition and continuous relaxation method are adopted. Zhang *et al.* [17] proposed a cloud-based MEC offloading framework in vehicular networks, where both the heterogeneous requirements of the mobility of the vehicles and the computation tasks are considered. Based on the analysis of the characteristics of various offloading strategies, the authors further propose a predictive-mode transmission scheme for task-file uploading. Moreover, Zhang *et al.* [8] adopted a Stackelberg game theory approach to design an optimal multilevel offloading scheme, which maximizes the utilities of both the vehicles and the computing servers.

Most of the existing MEC-based resource allocation and optimization problems are mixed-integer nonlinear programming (MINLP) problems and they are also NP-hard problems, which are not computable in polynomial time with existing general time with existing general solvers [18]. Generally speaking, the complex optimization problem can be decomposed into subproblems, and by solving the subproblems, respectively, the near-optimal solution is derived [7]–[9], [19], [20].

In order to efficiently solve the resource allocation optimization problem, some other methods, such as game theory, heuristic intelligent algorithm, and DRL, are also becoming the research focuses. Messous *et al.* [21] considered the problem of computation offloading while achieving a tradeoff between execution time and energy consumption in an unmanned aerial vehicle (UAV) network, where the combination of energy overhead and delay is minimized by the designed game theory model. Dinh *et al.* [22] formulated a distributed computation offloading problem among the mobile users as an exact potential game and proposed a distributed offloading scheme based on *Q*-learning and better response with inertia and prove the Nash equilibrium convergence.

As to DRL-based approach, Zhang *et al.* [2] adopted a deep *Q*-learning approach for designing an optimal data transmission scheduling scheme in cognitive vehicular networks to minimize transmission costs while also fully utilizing various communication modes and resources. In [16], they also design an optimal offloading scheme with joint MEC server selection and transmission mode determination in a deep *Q*-learning approach to maximize task offloading utility. Tan and Hu [14] developed a framework of joint optimal resource allocation of communication, caching, and computing for vehicular networks and proposed a DRL approach to solve this resource allocation problem. He *et al.* [15] proposed an integrated framework that can enable dynamic orchestration of networking, caching, and computing resources for connected vehicles and formulated the resource allocation strategy as a joint optimization problem. To derive the optimal strategy, they proposed a DDQN-based approach to solve the problem.

All those works above are marvelous solutions. Most existing relevant works focus on how to allocate resource of networking, caching, and computing for better data scheduling and processing. The data are generally offloaded to RSUs or cell towers to compute. Few works focus on the collaborative computing on the vehicle side. This article takes the research a step further by fully utilizing the idle computing resources of collaborative vehicles, formulating a unified framework with communication, computation, caching, and collaborative computing, and developing a collaborative data scheduling scheme to minimize the system-wide data processing cost with ensured delay constraints of applications.

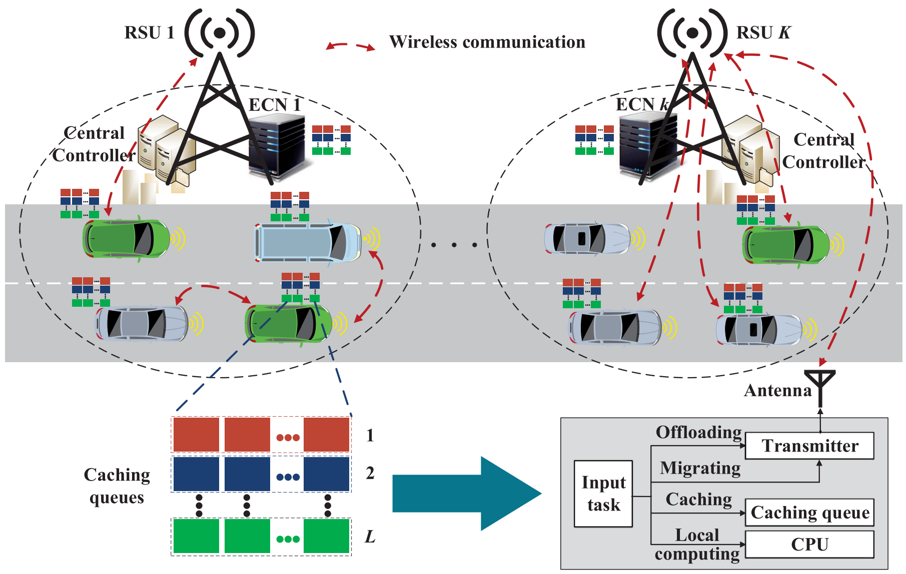

Fig. 1. Unified framework with communication, computation, caching, and collaborative computing in the VEC network.

III. System Model

This section presents the system model, including the unified framework with communication, computation, caching, and collaborative computing; and the multiqueue model for data caching on dual sides. For convenience, the main notations used are summarized in Table I.

A. Unified Framework With Communication, Computation, Caching, and Collaborative Computing in the VEC Network

Fig. 1 shows the unified framework with communication, computation, caching, and collaborative computing in the VEC network. The road is divided into K segments, denoted by $\mathbb{K} = \{1, 2, \ldots, K\}$, and each covered by an RSU with an ECN. The coverage radius of these RSUs is denoted by $\{R_1, R_2, \ldots, R_K\}$, respectively. Vehicles can establish the communication link with both RSU and vehicles through the assigned orthogonal licensed channels, each with a bandwidth of B. The licensed bandwidth is classified into two categories, one is for vehicle-to-infrastructure (V2I) communication, and the other for vehicle-to-vehicle (V2V) communication. The number of the licensed channel for V2I and V2V communications is denoted by C_{V2I} and C_{V2V}, respectively. RSUs also provide powerful computing capacity due to the deployed ECN.

In a VEC scenario, various data would be generated from onboard applications both for safety (e.g., high-definition camera and LiDAR) and entertainment (e.g., augmented reality and face recognition) purposes [23], [24]. These applications generally utilize the deep learning method to process data, which need powerful computational capacity [25]. In order to better describe the data generating, transmitting and computing processes, we divide time into time slots, each with a length of Δt. Since Δt is short, we consider the system to be quasistatic so that the wireless channels and the topology of the system keep unchanged at each time slot and vary at different time slots [19]. Data would be generated at the beginning of each time slot.

TABLE I
Major Notations

Notation	Explanation
K	Number of road segments
C_{V2I}	Number of licensed channels for V2I communication
C_{V2V}	Number of licensed channels for V2V communication
B	Bandwidth of each licensed channel
R_k	Coverage radius of RSU k
R^{V}	Coverage radius of vehicles
N_k	Number of vehicles within RSU k
Δt	Duration of a time-slot
M	Number of data types
D_i	Amount of type-i data
c	Processing density of data
f_n^{local}	Processing capability of vehicle n
f_k^{ECN}	Processing capability of RSU k
κ_1, κ_2	Effective switched capacitance related to the chip architecture in vehicles and RSUs
d	Distance between transmitter and receiver
ϑ	Path loss exponent
h	Channel fading coefficient
ω_0	White Gaussian noise power
P_n^{tr}	Transmission power of vehicle n
$\alpha_{n,k}^t, \beta_{n,k}^t$	Local computing and data offloading indicators for vehicle n running on road segment k at time-slot t
$\gamma_{n,k}^t, \delta_{n,k}^t$	Data migrating and receiving indicators for vehicle n running on road segment k at time-slot t
μ_n^t	Data processing indicator for RSU n at time-slot t
ξ	Penalty coefficient
$q_{n,l}^t, g_{k,l}^t$	Length of data cached in queue l of vehicle n and RSU k at time-slot t
$c^{\mathsf{I}}, c^{\mathsf{V}}$	Cost for using licensed V2I and V2V channels
c^{ECN}	Cost for RSU processing data
ϱ	Cost for energy consumption

Due to the limited computing capacity of the onboard device, severe delay would be caused for processing computation-intensive data. Thanks to the licensed channel resources, some latency-sensitive and computation-intensive data of vehicles could be offloaded to RSUs or migrated to collaborative vehicles that have idle computation resources. The control center in each road segment schedules vehicular communications through a dedicated control channel. Besides, the control center plays a role in gathering the states of vehicles through pilot signals.

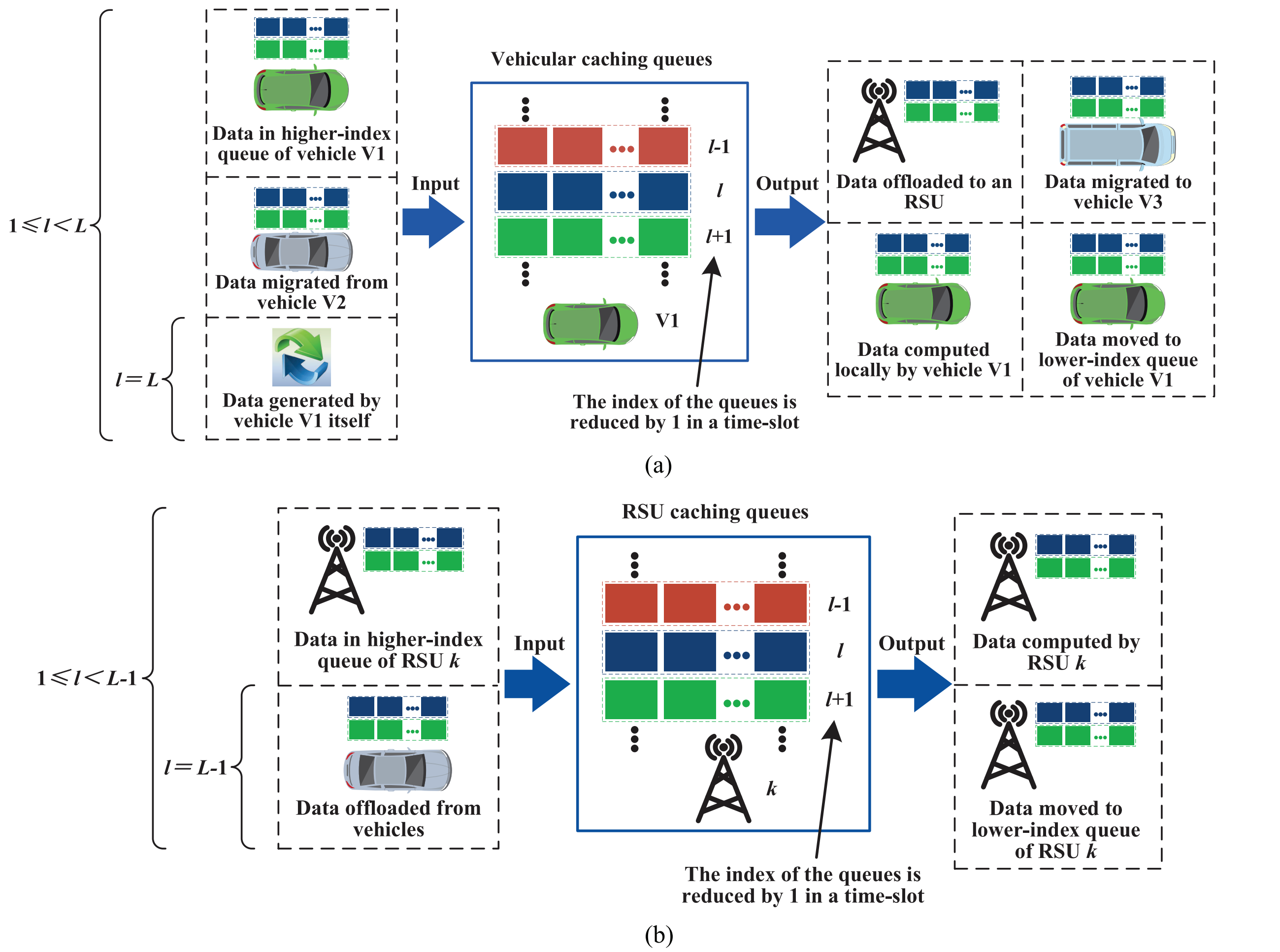

Fig. 2. Multiqueue model for caching data on dual sides. (a) Multiqueue model for data caching on vehicle side. According to the queue index, there are two cases, i.e., when $1 \leq l < L$ and when $l = L$. For $1 \leq l < L$, the input data come from three sources. For $l = L$, the input data only come from one source. (b) Multiqueue model for data caching on RSU side.

The size of the pilot signal can be set to arbitrarily small. Thus, the extra overhead can be very small [26]. Since the processing result is usually very tiny, we neglect the output return process and just focus on transmitting data to RSUs or collaborative vehicles [19]. We categorize data into M types and use two items $\{D_i, T_i\}$ to describe an arbitrary type of data i ($i \in \mathbb{M} = \{1, 2, \ldots, M\}$), where D_i stands for the amount of data, and T_i stands for the data delay constraint. In addition, we use c to represent the processing density (in CPU cycles/bit) of the data. It is noteworthy that the unit of delay constraint is time slot, which means once the data are generated, it needs to be processed within the time duration of $\Delta t \times T_i$. At each time slot, type-i data are generated with the probability of ϖ_i, which meets $\Sigma_{i=i}^{M} \varpi_i \leq 1$. The generated data can be processed locally, or offloaded to an RSU or migrated to a collaborative vehicle to be processed. For some delay-tolerant data, it may give way to the delay-sensitive data in the vehicular caching for priority processing. In order to better illustrate it, we model the data caching on both the vehicle side and RSU side as a multiqueue system in the following.

B. Multiqueue Model for Data Caching on Dual Sides

For the data caching on both the vehicle side and RSU side, we use L to denote the number of caching queues indexed as $\{1, 2, \ldots, L\}$, respectively. According to the delay constraints of generated data, we have $L = \max\{T_i, i \in \mathbb{M}\}$. In each queue, the remaining lifetime of data under its delay constraint is the same. Since the caching queue's input and output of vehicles and RSUs are different, we analyze them, respectively, in the following.

1) Vehicle Side: On the vehicle side as shown in Fig. 2(a), we divide the queue into two categories according to the queue index l, i.e., $1 \leq l < L$ and $l = L$. Assume the index of time slot now is t, for the case when $1 \leq l < L$, the input data in caching queue l in a given vehicle V1 come from three sources.

1) The data in $l+1$ of V1 and has not been processed at time slot $t-1$. As time slot increases from $t-1$ to t, the remaining lifetime for data processing decreases by 1.
2) The data generated by vehicle V1 itself at time slot t with data delay constraint l.
3) Another vehicle V2 transmitted data with queue index $l+1$ to V1 at time slot $t-1$.

The data in queue l of vehicle V1 when $1 \leq l < L$ has four different outputs.

1) Be offloaded to RSU k and then cached in queue $l-1$ of RSU k at next time slot $t+1$.
2) Be migrated to another vehicle V3 and then cached in $l-1$ of V3 at next time slot $t+1$.

3) Be computed locally through the computation resource of vehicle V1.
4) If none of the three actions above is adopted, data in queue l will be moved to queue $l-1$ of vehicle V1 if $l \neq 1$, otherwise deleted if $l = 1$, at next time slot $t+1$.

For the case when $l = L$, the input data in caching queue l only come from the newly generated data with delay constraint L. The output of data in queue l when $l = L$ has four transitions, the same as when $1 \leq l < L$.

2) RSU Side: On the RSU side, as shown in Fig. 2(b), we also detail the input and output of data in each queue l. We divide the queue into two categories according to the queue index, i.e., $1 \leq l < L-1$ and $l = L-1$. It is noteworthy that the maximal queue index l of RSUs is $L-1$. This is because data cannot be generated by RSUs themselves and it will take at least one-time slot for data to be offloaded from vehicles to RSUs. Just like the analysis on the vehicle side, we assume the index of time slot now is t, for the case when $1 \leq l < L-1$, the input data in caching queue l in a given RSU k comes from two sources.[1]

1) The data in $l+1$ of RSU k and has not been processed at time slot $t-1$.
2) Vehicles transmitted data with queue index $l+1$ to RSU k at time slot $t-1$.

The data in queue l of RSU k when $1 \leq l < L-1$ has two different outputs.

1) Be computed through the computation resource of ECN-enable RSU.
2) Be moved to queue $l-1$ of RSU k if $l \neq 1$, otherwise deleted if $l = 1$, at next time slot $t+1$.

For the case when $l = L-1$, the input data in caching queue l only consist of the data with remaining lifetime $l+1$ (i.e., data with remaining lifetime L) offloaded from vehicles through V2I communication at time slot $t-1$. The output of data in queue l when $l = L-1$ has two different outputs, the same as when $1 \leq l < L-1$.

IV. Data Scheduling: Analysis, Problem Formulation, and MDP

In this section, we first investigate the performance of the formulated unified VEC framework with various transmission and computation modes. Then, we formulate an optimal data scheduling problem that takes into account both cost and delay constraints.

A. Analysis of Data Scheduling: Transmission and Computation

The data cached in the caching queues of both vehicles and RSUs can be scheduled in different ways. We will detail them in the following.

1) Data Processed Locally: We denote the processing capability (i.e., the amount of CPU frequency in cycles/s) at vehicle n assigned for local computing as f_n^{local}, then the power consumption for vehicle n to process data locally is expressed as

$$p_n^{\text{local}} = \kappa_1 \left(f_n^{\text{local}}\right)^3 \tag{1}$$

where κ_1 stands for the effective switched capacitance related to the chip architecture in vehicle [27]. Accordingly, the energy consumption of vehicle n for local processing during one-time slot is expressed as

$$E_n^{\text{local}} = p_n^{\text{local}} \Delta t = \kappa_1 \left(f_n^{\text{local}}\right)^3 \Delta t. \tag{2}$$

The amount of data be processed locally by vehicle n during one-time slot is expressed as

$$D_n^{\text{local}} = \frac{f_n^{\text{local}} \Delta t}{c}. \tag{3}$$

2) Data Offloaded to RSU: In the VEC network, the data can be offloaded to RSU through V2I communication scheduled by the control center. To model the data offloading through V2I communication, we denote the path loss as $d^{-\vartheta}$, where d and ϑ denote the distance from the transmitter to the receiver and the path-loss exponent, respectively. Moreover, the channel fading coefficient is denoted by h, which is modeled as a circularly symmetric complex Gaussian random variable [28]. When data are offloaded from vehicle n to RSU k on a licensed V2I channel, the transmission rate is given by

$$r_{n,k}^{t,\text{V2I}} = B \log_2 \left(1 + \frac{P_n^{\text{tr}} |h|^2}{\omega_0 \left(d_{n,k}^t\right)^{\vartheta}} \right) \tag{4}$$

where P_n^{tr} is the transmission power of vehicle n, ω_0 denotes the white Gaussian noise power, and $d_{n,k}^t$ denotes the distance from vehicle n to RSU k at time slot t. The energy consumption of vehicle n for transmitting data during one-time slot is expressed as

$$E_n^{\text{tr}} = P_n^{\text{tr}} \Delta t. \tag{5}$$

3) Data Processed by RSU: The data in the caching queues of RSUs are processed by deployed ECNs. We denote the processing capability at RSU k as f_k^{ECN}, then the power consumption for RSU k to process data is expressed as

$$p_k^{\text{ECN}} = \kappa_2 \left(f_k^{\text{ECN}}\right)^3 \tag{6}$$

where κ_2 stands for the effective switched capacitance related to the chip architecture in RSU [27]. Accordingly, the energy consumption of RSU k for data processing during one-time slot is expressed as

$$E_k^{\text{ECN}} = p_k^{\text{ECN}} \Delta t = \kappa_2 \left(f_k^{\text{ECN}}\right)^3 \Delta t. \tag{7}$$

The amount of data processed by RSU k during one-time slot is expressed as

$$D_k^{\text{ECN}} = \frac{f_k^{\text{ECN}} \Delta t}{c}. \tag{8}$$

[1]In this article, we do not consider the migration of data between RSUs, which will be considered as our future work.

4) Data Migrated to Collaborative Vehicle: When some vehicles have no data to process or their data in caching queues are delay tolerant and have longer remaining lifetime, they can aid data processing as collaborative vehicles. In this way, data can be also migrated to collaborative vehicles through V2V communication. We denote the communication radius of vehicles as R^{V}. For arbitrary vehicle n and collaborative vehicle n', n can transmit data to n' while n does not take any transmission action at the same time slot. The communication model is similar to the V2I communication model and (4). When data are migrated from n to n' on a licensed V2V channel at time slot t, the transmission rate is given by

$$r_{n,n'}^{t,\mathsf{V2V}} = B\log_2\left(1 + \frac{P_n^{\mathsf{tr}}|h|^2}{\omega_0\left(d_{n,n'}^t\right)^{\vartheta}}\right) \tag{9}$$

where $d_{n,n'}^t$ is the distance between vehicle n and n'. It is noteworthy that only when $d_{n,n'}^t < R^{\mathsf{V}}$ does (9) hold.

B. Problem Formulation

At a give time slot, the data of each vehicle can be: 1) processed locally; 2) offloaded to RSUs; 3) migrated to collaborative vehicles; and 4) kept in its caching queues. Let $\alpha_{n,k}^t = 1$ indicate that the data of vehicle n running on road segment k at time slot t are processed locally. Similarly, we use $\beta_{n,k}^t = 1$ to indicate that the data of vehicle n are offloaded to RSU through V2I communication, and $\gamma_{n,k}^t = 1$ to indicate that the data of vehicle n are migrated to a collaborative vehicle through V2V communication. The case $\alpha_{n,k}^t = \beta_{n,k}^t = \gamma_{n,k}^t = 0$ indicates that vehicle n keeps the data in its caching queues. In addition, vehicle n can receive data migrated from other vehicles, let $\delta_{n,k}^t$ indicate that vehicle n on road segment k is on the receiving mode at time slot t. On the RSU side, each RSU can process data or keep the data in its caching queues. We use $\mu_k^t = 1$ to indicate that RSU k processes data at time slot t, and $\mu_k^t = 0$ to indicate that RSU k keeps the data in its caching queues.

To promote data processing, we introduce a penalty mechanism, which means a penalty will be resulted in if data are not processed before its deadline. We use ξ to denote the penalty coefficient indicating the penalty amount of one unit data. Moreover, costs for communication and computation are also produced during data scheduling. Those costs include: 1) the cost for using licensed channels of V2I and V2V communications; 2) the cost for RSU computing data; and 3) the energy consumption cost during data computing and transmitting. To be able to process data under delay constraints, the data with a smaller queue index should be processed first. We use $\mathcal{V}_k^t$ to denote the vehicle set in road segment k at time slot t. The amount of data not meeting the delay constraints at time slot t is then expressed as

$$D_{\mathsf{loss}}^t = \sum_{k\in\mathbb{K}}\left(\sum_{n\in\mathcal{V}_k^t}\alpha_{n,k}^t\max\left\{0, q_{n,1}^t - D_n^{\mathsf{local}}\right\} + \mu_k^t\max\left\{0, g_{k,1}^t - D_k^{\mathsf{ECN}}\right\} + \left(1-\alpha_{n,k}^t\right)q_{n,1}^t + \left(1-\mu_k^t\right)g_{k,1}^t\right) \tag{10}$$

where $q_{n,1}^t$ and $g_{k,1}^t$ denote the amount of data cached in queue 1 of vehicle n and RSU k at time slot t, respectively. D_n^{local} and D_k^{ECN} are the data processing capabilities of vehicles and RSUs at one-time slot, as formulated by (3) and (8), respectively.

To reduce cost while ensuring delay constraints through making full use of communication and computation resources, we formulate the data scheduling problem as

$$\begin{aligned}\min_{\{\alpha,\beta,\gamma,\delta,\mu\}}\ \text{Loss} = \sum_{t=1}^{\infty}\Bigg\{&\xi D_{\mathsf{loss}}^t + \sum_{k\in\mathbb{K}}\Bigg(\varrho E_n^{\mathsf{local}}\sum_{n\in\mathcal{V}_k^t}\alpha_{n,k}^t\\ &+ \left(c^{\mathsf{I}} + \varrho E_n^{\mathsf{tr}}\right)\sum_{n\in\mathcal{V}_k^t}\beta_{n,k}^t\left(1-\delta_{n,k}^t\right)\\ &+ \left(c^{\mathsf{V}} + \varrho E_n^{\mathsf{tr}}\right)\sum_{n\in\mathcal{V}_k^t}\gamma_{n,k}^t\left(1-\delta_{n,k}^t\right)\\ &+ \left(c^{\mathsf{ECN}} + \varrho E_k^{\mathsf{ECN}}\right)\mu_k^t\Bigg)\Bigg\}\\ \text{s.t.}\quad &\text{C1: } \alpha_{n,k}^t,\ \beta_{n,k}^t,\ \gamma_{n,k}^t,\ \delta_{n,k}^t,\ \mu_k^t \in \{0,1\}\\ &\text{C2: } \beta_{n,k}^t\gamma_{n,k}^t = 0\\ &\text{C3: } \beta_{n,k}^t\delta_{n,k}^t = \gamma_{n,k}^t\delta_{n,k}^t = 0\\ &\text{C4: } \alpha_{n,k}^t\beta_{n,k}^t = \alpha_{n,k}^t\gamma_{n,k}^t = 0\end{aligned} \tag{11}$$

where ϱ is a weight coefficient indicating the energy consumption cost of one unit energy during data computing and transmitting [29], and c^{I} and c^{V} denote the costs at a time slot for using licensed channels for V2I and V2V communications, respectively. c^{ECN} denotes the cost for RSU processing data at a time slot. In (11), constraint C1 indicates that whether a vehicle or an RSU takes one action or not at time slot t. Constraint C2 indicates that vehicles cannot transmit data through V2I and V2V communications simultaneously. Constraint C3 indicates that when a vehicle is on the receiving mode, it cannot transmit data neither through V2I nor V2V communication. Constraint C4 indicates that at most one data processing action among local computing, offloading through V2I, and migrating through V2V can be chosen during one-time slot.

C. Model Data Processing Scheduling as MDP

In (11), *Loss* mainly depends on the states and data scheduling actions of both vehicles and RSUs. States of the next time slot only depend on the current state and the data scheduling actions. Accordingly, the data scheduling problem can be formulated as a Markov decision process (MDP) to analyze the state transitions of caching queues.

We denote the state of the MDP at time slot t as $S^t \triangleq \{S_1^t, S_2^t, \ldots, S_K^t, \Phi^t\}$, where Φ^t is the position state of vehicles and RSUs, and S_k^t $(1 \le k \le K)$ is the caching state of vehicles and RSU in road segment k. Since the positions of RSUs are fixed and the next positions of vehicles can be obtained based on the current positions and running speeds, the next state Φ^{t+1} can be easily obtained. Therefore, we mainly focus on the caching state transitions. In arbitrary road segment k, we define $S_k^t \triangleq \{Q_{1,k}^t, Q_{2,k}^t, \ldots, Q_{|\mathcal{V}_k^t|,k}^t, G_k^t\}$, where $|\mathcal{V}_k^t|$ is the number of vehicles in road segment k at time slot t, $Q_{n,k}^t$ ($n \in \mathcal{V}_k^t$) is the caching state of vehicle n and is expressed as $\{q_{n,1}^t, q_{n,2}^t, \ldots, q_{n,L}^t\}$, G_k^t is the caching state of RSU k and is expressed as $\{g_{k,1}^t, g_{k,2}^t, \ldots, g_{k,L}^t\}$. $q_{n,l}^t$, and $g_{k,l}^t$ ($1 \le l \le L$) denote the amount of data cached in queue l of vehicle n and RSU k at time slot t, respectively. We define the action taken by RSUs and vehicles as $A^t \triangleq \{A_1^t, A_2^t, \ldots, A_K^t\}$. For the RSU and vehicles in road segment k, we define $A_k^t \triangleq \{a_{1,k}^t, a_{2,k}^t, \ldots, a_{|\mathcal{V}_k^t|,k}^t, \tilde{a}_k^t\}$, where $a_{n,k}^t$ ($n \in \mathcal{V}_k^t$) and $\tilde{a}_k^t$ are the actions of vehicle n and RSU k, respectively. $a_{n,k}^t$ is consisted of the possible data scheduling actions, expressed as $a_{n,k}^t = \{\alpha_{n,k}^t, \beta_{n,k}^t, \gamma_{n,k}^t, \delta_{n,k}^t\}$. $\tilde{a}_k^t$ can be expressed as $\{\mu_k^t\}$. To describe the caching state at the next time slot $t+1$, we should calculate the amount of data transmission during the current time slot t. The amount of data that is transmitted from vehicle n to RSU k can be expressed as

$$D_{n,k}^{t,\mathsf{off}} = \begin{cases} \dfrac{\beta_{n,k}^t r_{n,k}^{t,\mathsf{V2I}} \Delta t C_{\mathsf{V2I}}}{\left|\mathcal{V}_k^{t,\mathsf{off}}\right|}, & \mathcal{V}_k^{t,\mathsf{off}} \neq \varnothing \\ 0, & \mathcal{V}_k^{t,\mathsf{off}} = \varnothing \end{cases} \tag{12}$$

where $\mathcal{V}_k^{t,\mathsf{off}}$ denotes the set of vehicles in road segment k choosing to offload data to RSU at time slot t. In addition, we use $\mathcal{V}_k^{t,\mathsf{mig}}$ and $\mathcal{V}_k^{t,\mathsf{rec}}$ to denote the sets of vehicles in road segment k choosing to transmit data to collaborative vehicles and choosing to receive migration data at time slot t, respectively. We define an indicator $I_{n,n'}^{t,k}$ to denote the connection between vehicle n ($n \in \mathcal{V}_k^{t,\mathsf{mig}}$) and n' ($n' \in \mathcal{V}_k^{t,\mathsf{rec}}$), which equals 1 when the connection is established and 0 otherwise, $I_{n,n'}^{t,k}$ is defined as

$$I_{n,n'}^{t,k} = \begin{cases} 1, & \gamma_{n,k}^t = 1,\ \delta_{n',k}^t = 1,\ d_{n,n'} < R^{\mathsf{V}} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

We assume that a vehicle n can only transmit data to at most one collaborative vehicle at one-time slot, hence $I_{n,n'}^{t,k}$ meets the condition that $\sum_{n' \in \mathcal{V}_k^{t,\mathsf{rec}}} I_{n,n'}^{t,k} \le 1$. The amount of data that is transmitted from vehicle n to n' can be expressed as

$$D_{n,n',k}^{t,\mathsf{mig}} = \begin{cases} I_{n,n'}^{t,k} \dfrac{r_{n,n'}^{t,\mathsf{V2V}} \Delta t C_{\mathsf{V2V}}}{\left|\mathcal{V}_k^{t,\mathsf{mig}}\right|}, & \mathcal{V}_k^{t,\mathsf{mig}} \neq \varnothing \\ 0, & \mathcal{V}_k^{t,\mathsf{mig}} = \varnothing. \end{cases} \tag{14}$$

We use $l_{\mathsf{min}}^{t,n}$ to denote the smallest index of the queue with nonempty queueing data of arbitrary vehicle n at time slot t. Then, the amount of data migrated from queue $l+1$ of vehicle $\bar{n}$ to queue l of vehicle n and RSU k at time slot t are expressed as

$$D_{\bar{n},n,k,l}^{t,\mathsf{V2V}} = \begin{cases} \mathbf{1}\Big(l_{\mathsf{min}}^{t,\bar{n}} == l+1\Big) & \\ \quad \times\ I_{\bar{n},n}^{t,k} r_{\bar{n},n}^{t,\mathsf{V2V}} \Delta t C_{\mathsf{V2V}} / |\mathcal{V}_k^{t,\mathsf{mig}}|, & \mathcal{V}_k^{t,\mathsf{mig}} \neq \varnothing \\ 0, & \mathcal{V}_k^{t,\mathsf{mig}} = \varnothing \end{cases} \tag{15}$$

and

$$D_{\bar{n},k,l}^{t,\mathsf{V2I}} = \begin{cases} \mathbf{1}\Big(l_{\mathsf{min}}^{t,\bar{n}} == l+1\Big) & \\ \quad \times\ \beta_{\bar{n},k}^t r_{\bar{n},k}^{t,\mathsf{V2I}} \Delta t C_{\mathsf{V2I}} / |\mathcal{V}_k^{t,\mathsf{off}}|, & \mathcal{V}_k^{t,\mathsf{off}} \neq \varnothing \\ 0, & \mathcal{V}_k^{t,\mathsf{off}} = \varnothing \end{cases} \tag{16}$$

respectively, where $\mathbf{1}(\tau)$ is an indicator function which equals 1 if τ is true and 0 otherwise. Accordingly, given caching state $Q_{n,k}^t$ of vehicle n in road segment k, the state of queues that form $Q_{n,k}^{t+1}$ is consisted of two parts, namely, $q_{n,l_{\mathsf{min}}^{t,n}}^{t+1}$ and $q_{n,l}^{t+1}$ ($l \neq l_{\mathsf{min}}^{t,n}$), which are expressed as

$$q_{n,l_{\mathsf{min}}^{t,n}}^{t+1} = \begin{cases} \max\Big\{0, q_{n,2}^t + \varpi_1 D_1 & \\ \quad + \sum_{\bar{n} \in \mathcal{V}_k^{t,\mathsf{mig}}} D_{\bar{n},n,k,1}^{t,\mathsf{V2V}} & \\ \quad - D_n^{\mathsf{local}}\Big\}, & l_{\mathsf{min}}^{t,n} = 1 \\ \max\Big\{0, q_{n,l_{\mathsf{min}}^{t,n}+1}^t + \varpi_{l_{\mathsf{min}}^{t,n}} D_{l_{\mathsf{min}}^{t,n}} & \\ \quad + \sum_{\bar{n} \in \mathcal{V}_k^{t,\mathsf{mig}}} D_{\bar{n},n,k,l_{\mathsf{min}}^{t,n}}^{t,\mathsf{V2V}} & \\ \quad - \sum_{n' \in \mathcal{V}_k^{t,\mathsf{rec}}} D_{n,n',k}^{t,\mathsf{mig}} & \\ \quad - D_{n,k}^{t,\mathsf{off}} - D_n^{\mathsf{local}}\Big\}, & 1 < l_{\mathsf{min}}^{t,n} < L \\ \max\Big\{0, \varpi_L D_L - D_{n,k}^{t,\mathsf{off}} - D_n^{\mathsf{local}} & \\ \quad - \sum_{n' \in \mathcal{V}_k^{t,\mathsf{rec}}} D_{n,n',k}^{t,\mathsf{mig}}\Big\}, & l_{\mathsf{min}}^{t,n} = L \end{cases} \tag{17}$$

and

$$q_{n,l}^{t+1} = \begin{cases} q_{n,l+1}^t + \varpi_l D_l & \\ \quad + \sum_{\bar{n} \in \mathcal{V}_k^{t,\mathsf{mig}}} D_{\bar{n},n,k,l}^{t,\mathsf{V2V}}, & l \neq l_{\mathsf{min}}^{t,n}, 1 \le l < L \\ \varpi_l D_l, & l \neq l_{\mathsf{min}}^{t,n}, l = L \end{cases} \tag{18}$$

respectively. Similarly, we use $\tilde{l}_{\mathsf{min}}^{t,k}$ to denote the smallest index of the queue with nonempty queueing data of arbitrary RSU k at time slot t. Based on caching state G_k^t of RSU k, the state of queues that form G_k^{t+1} is consisted of two parts, namely, $g_{k,\tilde{l}_{\mathsf{min}}^{t,k}}$ and $g_{k,l}$ ($l \neq \tilde{l}_{\mathsf{min}}^{t,k}$), which are expressed as

$$g_{k,\tilde{l}_{\mathsf{min}}^{t,k}}^{t+1} = \begin{cases} \max\Big\{0, g_{k,\tilde{l}_{\mathsf{min}}^{t,k}+1}^t - D_k^{\mathsf{ECN}}\Big\} & \\ \quad + \sum_{\bar{n} \in \mathcal{V}_k^{t,\mathsf{off}}} D_{\bar{n},k,\tilde{l}_{\mathsf{min}}^{t,k}}^{t,\mathsf{V2I}}, & 1 \le \tilde{l}_{\mathsf{min}}^{t,k} < L-1 \\ \sum_{\bar{n} \in \hat{\mathcal{V}}_k^{t,\mathsf{off}}} D_{\bar{n},k,\tilde{l}_{\mathsf{min}}^{t,k}}^{t,\mathsf{V2I}} - D_k^{\mathsf{ECN}}, & \tilde{l}_{\mathsf{min}}^{t,k} = L-1 \end{cases} \tag{19}$$

and

$$g_{k,l}^{t+1} = \begin{cases} g_{k,l+1}^t + \sum_{\bar{n} \in \hat{\mathcal{V}}_k^{t,\mathsf{off}}} D_{\bar{n},k,l}^{t,\mathsf{V2I}}, & l \neq \tilde{l}_{\mathsf{min}}^{t,k},\ 1 \le l < L-1 \\ \sum_{\bar{n} \in \hat{\mathcal{V}}_k^{t,\mathsf{off}}} D_{\bar{n},k,l}^{t,\mathsf{V2I}}, & l \neq l_{\mathsf{min}}^{t,k},\ l = L-1 \end{cases} \tag{20}$$

respectively.

In state S^t, the penalty and cost by taking action A^t can be expressed as

$$\begin{aligned}\mathrm{Loss}^t = \xi D^t_{\mathrm{loss}} + \sum_{k\in\mathbb{K}}\Bigg(&\varrho E_n^{\mathrm{local}}\sum_{n\in\mathcal{V}_k^t}\alpha^t_{n,k}\\ &+\left(c^{\mathrm{I}}+\varrho E_n^{\mathrm{tr}}\right)\sum_{n\in\mathcal{V}_k^t}\beta^t_{n,k}\left(1-\delta^t_{n,k}\right)\\ &+\left(c^{\mathrm{V}}+\varrho E_n^{\mathrm{tr}}\right)\sum_{n\in\mathcal{V}_k^t}\gamma^t_{n,k}\left(1-\delta^t_{n,k}\right)\\ &+\left(c^{\mathrm{ECN}}+\varrho E_k^{\mathrm{ECN}}\right)\mu_k^t\Bigg). \quad (21)\end{aligned}$$

The purpose of the MDP is deriving an optimal data scheduling strategy that minimizes the cumulative value of Loss^t over time slots. The optimal strategy that indicates the data scheduling actions, is expressed as

$$\pi^* = \arg\min_{\pi}\ \mathrm{E}\left(\sum_{t=1}^{\infty}\eta^t\mathrm{Loss}^t\right) \quad (22)$$

where $0 < \eta < 1$ is a discount factor used to indicate the impact of future *Loss* on current actions.

V. DQN-Based Optimal Data Scheduling Scheme

A. From Q-Learning to the Deep Q-Network

In the formulated MDP problem (22), the large scale of state space and action space makes it hard to find the optimal data scheduling strategy π^* [30]. Fortunately, the reinforcement learning technology is powerful in handling the scheduling problem [13]. Reinforcement learning is a main branch of machine learning, where agents learn to take series of actions that maximize the cumulative future reward with corresponding policies over states [15], [16]. Accordingly, our proposed MDP can be considered as a reinforcement learning problem, to minimize cumulative future loss. Under a given data scheduling strategy π, the expected long-term loss from taken action A^t at state S^t can be expressed as an action-value function (i.e., Q-function), which is shown as

$$\begin{aligned}Q_\pi\left(S^t,A^t\right) &= \mathrm{E}\left[\sum_{i=0}^{\infty}\eta^i\mathrm{Loss}^{t+i}\middle|\left(S^t,A^t\right)\right]\\ &= \mathrm{E}\left[\mathrm{Loss}^t+\eta^1\mathrm{Loss}^{t+1}+\cdots\middle|\left(S^t,A^t\right)\right]\\ &= \mathrm{E}_{S^{t+1}}\left[\mathrm{Loss}^t+\eta Q_\pi\left(S^{t+1},A^{t+1}\right)\middle|\left(S^t,A^t\right)\right]. \quad (23)\end{aligned}$$

When given action A^t in state S^t, the expected minimum loss is expressed as

$$Q^*\left(S^t,A^t\right) = \mathrm{E}_{S^{t+1}}\left[\mathrm{Loss}^t+\eta\min_{A^{t+1}}Q\left(S^{t+1},A^{t+1}\right)\middle|\left(S^t,A^t\right)\right]. \quad (24)$$

Based on (24), the minimum $Q^*(S^t, A^t)$ and optimal data scheduling actions can be derived by value and action iteration. The updated process of $Q(S^t, A^t)$, namely, the Q-learning process, is expressed as

$$\begin{aligned}Q\left(S^t,A^t\right) \leftarrow Q\left(S^t,A^t\right)+\varphi\Big[&\mathrm{Loss}^t+\eta\min_{A^{t+1}}Q\left(S^{t+1},A^{t+1}\right)\\ &-Q\left(S^t,A^t\right)\Big] \quad (25)\end{aligned}$$

where φ is the learning rate.

Since a Q-table is used in the Q-learning process to store learned state–action combinations and corresponding Q-values, discrete state space is utilized in the Q-table. However, the states of the VEC network consist of the amount of data cached in the queues of vehicles and RSUs, whose value is continuous. Thus, the Q-learning approach cannot be directly implemented in solving our proposed MDP problem. To compensate the limitation of Q-learning, we incorporate the deep learning technology with the Q-learning method, which forms the DQN. Instead of Q-function, DQN uses a deep neural network as a nonlinear approximator that is able to capture the complex interaction among various states and actions. The inputs of the deep neural network are states, and the outputs are Q-values of actions. With the help of DQN, the Q-value in (23) can be estimated as $Q(S^t, A^t) \approx Q(S^t, A^t; \theta)$, where θ are the weights of the DQN. Accordingly, the optimal action for data scheduling in state S^t is the one with the minimum $Q(S^t, A^t; \theta)$, which is shown as

$$A^{t*} = \arg\min_{A^t} Q\left(S^t,A^t;\theta\right). \quad (26)$$

B. DQN Training

To guarantee the approximation ability of the estimated Q-value, $Q(S^t, A^t; \theta)$ should be trained toward the target value $\mathrm{Loss}^t + \eta\min_{A^{t+1}} Q(S^{t+1}, A^{t+1})$, which is substituted with the approximate target value as

$$y^t = \mathrm{Loss}^t+\eta\min_{A^{t+1}}Q\left(S^{t+1},A^{t+1};\theta^{t-1}\right). \quad (27)$$

To minimize the difference between estimated and target values, we define a loss function as

$$\mathrm{Er}\left(\theta^t\right) = \mathrm{E}\left[\left(y^t-Q\left(S^t,A^t;\theta^t\right)\right)^2\right] \quad (28)$$

where θ^t denotes the weights of the DQN at time slot t. The parameters from the previous time slot θ^{t-1} are held fixed when optimizing the loss function $Er(\theta^t)$ [31]. Differentiating the loss function with respect to the weights, we get the gradient as

$$\nabla_{\theta^t}Er(\theta^t) = \mathrm{E}\left[2\left(Q\left(S^t,A^t;\theta^t\right)-y^t\right)\nabla_{\theta^t}Q\left(S^t,A^t;\theta^t\right)\right]. \quad (29)$$

Based on the gradient descent, θ^t is updated as

$$\theta^t \leftarrow \theta^t - \varepsilon\nabla_{\theta^t}\mathrm{Er}\left(\theta^t\right) \quad (30)$$

where ε is a step size coefficient, which controls the updating step size in each iteration.

In order to improve learning efficiency while removing the correlations in the subsequent training samples at the same time, the experience replay technique is utilized in the learning process. The learned experience $e^t = (S^t, A^t, \mathrm{Loss}^t, S^{t+1})$ at

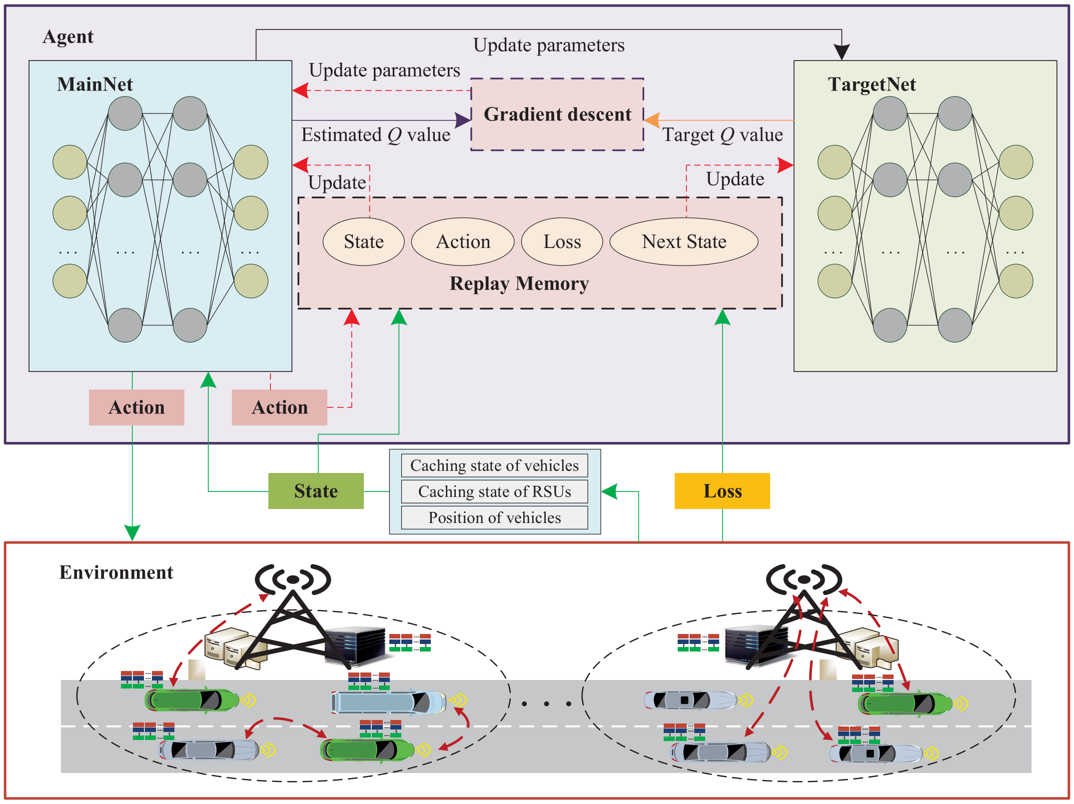

Fig. 3. DQN architecture for solving the data scheduling problem.

each time slot is stored in a data set $\mathcal{D}$ in a replay memory [32]. Then, we randomly draw a batch of stored experience as samples to train parameters of DQN. Before performing experience replay, an action is selected and executed according to an ϵ-greedy policy, which avoids local optimum while balancing exploration and exploitation during training. That is, a random action is chosen with probability ϵ to explore better data scheduling strategies, otherwise, the action that has the minimum Q-value is chosen.

It is noteworthy that the same parameters are used for calculating the estimated and target Q-values. As a consequence, there is a big correlation between the estimated and target Q-values. Therefore, it means that at every step of training, our estimated Q-value shifts but also the target Q-value shifts, which lead to a big oscillation in training. To address this issue, we introduce a separate target Q-network to calculate the target Q-value. The parameters of the target Q-network at time slot t are denoted as $\bar{\theta}^t$. The target Q-value is correspondingly expressed as

$$\bar{y}^t = \text{Loss}^t + \eta \min_{A^{t+1}} Q\left(S^{t+1}, A^{t+1}; \bar{\theta}^t\right). \tag{31}$$

Similarly, y^t in (28) and (29) is substituted by $\bar{y}^t$. $\bar{\theta}^t$ is hold fixed and only updated with the DQN parameters (θ^t) every ζ time slots. Fig. 3 shows the DQN architecture for solving the data scheduling problem in the VEC network, where MainNet refers to the neural network used to calculate the estimated Q-value and TargetNet refers to the neural network used to calculate the target Q-value. The full algorithm for training the DQN for optimal task scheduling is presented in Algorithm 1.

VI. Simulation Results and Discussions

In this section, we conduct simulations to validate the performance of the proposed data scheduling scheme. First, we describe the simulation scenario and parameter settings. Next, we discuss the simulation results.

A. Simulation Setup

We consider a two-way three-lane scenario. The length of per lane is 1000 m and the width of per lane is 4 m. One RSU is deployed in the middle of the roadside, with coordinate (0 m, 0 m). Three vehicles that are on three different lanes are keeping moving back and forth along their lanes. The initial positions of the three vehicles are set to (−500 m, 2 m), (0 m, 6 m), and (500 m, 10 m), respectively. For the speed of the three vehicles, we use part of the GAIA Open data set containing speeds of DiDi Express in Xi'an China [33]. The data set contains the GPS coordinates and real-time speeds of DiDi Express over 30 days and over thousands of districts and roads. We randomly choose three loads and calculate the average speeds of vehicles over the three roads, respectively. Based on the speed statistics, we set the speeds of the three vehicles to 17.7, 35.8, and 52.6 km/h, respectively. The coverage radius of RSU and vehicles is set to 500 and 250 m, respectively. The generated data of the vehicles are classified into four types. The length of the time slot is set to 100 ms. The data size is randomly distributed between 0.2 and 5.0, and the data delay constraints are randomly generated from {1, 2, 3, 4}. The detailed parameters setting about vehicles and RSU are shown in Table II.

Algorithm 1 DQN-Based Data Scheduling

Initialize replay memory $\mathcal{D}$
Initialize DQN with random weights θ
Initialize target DQN with weights $\bar{\theta} = \theta$
for episode $e = 1, ..., e_{\text{max}}$ **do**
  Observe the initial state S^1;
  for time-slot $t = 1, ..., t_{\text{max}}$ **do**
    Choose a random probability p;
    if $p \leq \epsilon$ **then**
      Select a random action A^t,
    else
      Choose action $A^t = \arg\min_{A^t} Q(S^t, A^t; \theta^t)$;
    end if
    Execute action A^t, calculate $Loss^t$ and derive the next state S^{t+1} according to formulas (17)–(20);
    Store the experience $(S^t, A^t, Loss^t, S^{t+1})$ into $\mathcal{D}$ in the replay memory;
    Get random minibatch of samples $(S^i, A^i, Loss^i, S^{i+1})$ from $\mathcal{D}$;
    Calculate the target Q-value from the target DQN, $\bar{y}^i = Loss^i + \eta \min_{A^{i+1}} Q(S^i, A^i; \bar{\theta}^t)$;
    Perform the gradient descent step on $Er(\theta^i) = \mathrm{E}[(\bar{y}^i - Q(S^i, A^i; \theta^t))^2]$ with respect to θ^t, and update θ^t;
    Every ζ time-slots, update $\bar{\theta}^t$ with θ^t;
  end for
end for

TABLE II
PARAMETERS SETTING ABOUT VEHICLES AND RSU

Description	Parameter	Value
Coverage radius of vehicles	R^{V}	250 m
Coverage radius of RSU	R_k	500 m
Length of time-slot	Δt	100 ms
Number of data types	M	4
Amount of type-i data	D_i	$0.2 \sim 5.0$
Delay constraints of type-i data	T_i	$\{1, 2, 3, 4\}$
Bandwidth per channel	B	1 MHz
Number of V2I channels	C_{V2I}	20
Number of V2V channels	C_{V2V}	10
Transmission power of vehicles	P_n^{tr}	30 dBm
White Gaussian noise power	ω_0	-100 dBm
Path loss exponent	ϑ	4
Switched capacitance coefficient	κ_1, κ_2	10^{-24}, 10^{-29}
Processing capability of RSU	f_k^{ECN}	100 G cycles/s
Processing capability of vehicle	f_n^{local}	1.4 G cycles/s
Processing density of data	c	100 cycles/bit
Penalty coefficient	ξ	5
Energy consumption cost coefficient	ϱ	2.44×10^{-4}
V2I cost coefficient	c^{I}	1
V2V cost coefficient	c^{V}	0.5
Cost for RSU processing data	c^{ECN}	2

Hereinafter, we introduce the simulation settings about DQN. The simulation uses TensorFlow [34] to implement the DQN. We use a GPU-based server with four NVIDIA GTX 2080 Ti GPUs, where the CPU is Intel Xeno E5-2690 v4 with 64-GB memory. The software environment we utilize is Keras 2.3.1 (based on TensorFlow) with Python 3.7 on Ubuntu 16.04.6 LTS. Both the main and target Q-networks use a fully connected deep neural network with two hidden layers. The

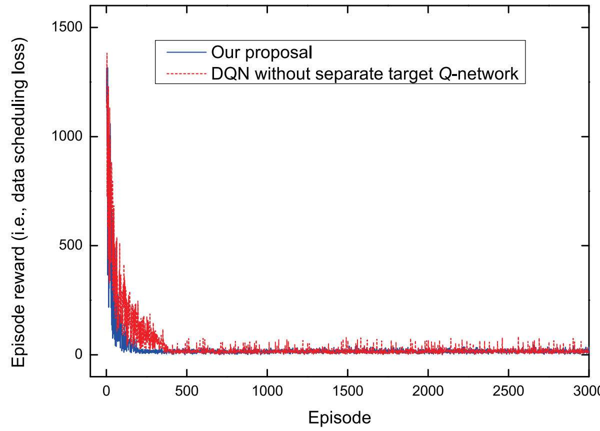

Fig. 4. Episode reward (i.e., data scheduling loss) achieved during training.

ReLU function is adopted as the activation function for hidden layers. The number of nodes for the hidden layers is set to 128. We set the learning rate and discount factor to 0.001 and 0.95, respectively. For the ϵ-greedy policy, we initialize $\epsilon = 1.0$ and let it decrease by a decay coefficient 0.995 over time slots until it reaches 0.1. The minibatch and maximum replay memory sizes are set to 64 and 10 000, respectively. The maximum time slots t_{max} and episodes e_{max} are set to 600 and 3000, respectively. The parameter updating frequency for target DQN ζ is set to 10. The detailed parameter setting of DQN is listed in Table III.

B. Simulation Results

1) Effectiveness: We mainly consider the following strategies and evaluate their performance under the same conditions.

1) Our proposal, that is, the proposed data scheduling scheme aiming at minimizing the cumulative value of loss over time slots, and it is solved by DQN with separated target Q-network (recorded as *our proposal*).
2) The proposed data scheduling scheme aiming at minimizing the cumulative value of loss over time slots, and it is solved by DQN without separate target Q-network (recorded as *DQN without separate target Q-network*).

We first evaluate the timeliness of *our proposal*. Similarly, the scheduling strategy *without separate target Q-network* is evaluated under the same conditions as a comparison scheme. Fig. 4 shows the changes in the reward (i.e., the data scheduling loss) obtained by the DRL agent and training episodes under different schemes. The figure shows that the loss of *our proposal* tends to be optimal and stable in about 200 episodes of training, and no divergence and oscillation problems are encountered in the simulation. As the episodes increase, the loss gradually stabilizes at the optimal value, which means that the agent of *our proposal* has learned the optimal data scheduling strategy to minimize long-term loss. The scheme *DQN without separate target Q-network* requires longer episodes to stabilize the loss (approximately 400). Although being stable, the loss of the scheme *DQN without separate target Q-network* is prone to relatively fluctuations and oscillation. This is because there is a big correlation between the estimated and target Q-values, and the estimated Q-values shift

TABLE III
PARAMETERS SETTING OF DQN

Description	Parameter	Value
State vector	S^t	$\{Q_1^t, Q_2^t, Q_3^t, G^t, \Phi^t\}$
Action vector	A^t	$\{a_1^t, a_2^t, a_3^t, \tilde{a}^t\}$
Network architecture		Fully connected neural network
Number of hidden layers		2
Number of nodes for the first hidden layer		128
Number of nodes for the second hidden layer		128
Activation function for hidden layers		ReLU
Learning rate		0.001
Discount factor	η	0.95
Initial ϵ		1
ϵ decay coefficient		0.995
Minimal ϵ		0.1
Minibatch size		64
Maximum replay memory size	$\lvert\mathcal{D}\rvert$	10000
maximum time-slots	$t_{\max}$	600
Maximum episodes	$e_{\max}$	3000
Parameter updating frequency for target DQN	ζ	10

but also the target Q-values shifts when the same parameters are used during the training process. The figure indicates that *our proposal* is more effective for solving the data scheduling problem.

2) Average Loss: Average loss indicates the average value of long-term data scheduling loss [i.e., *Loss* in (11)] over time slots. The following strategies are used for performance comparison.

1) The proposed data scheduling scheme based on DQN with separate target Q-network (record as *our proposal*).
2) The data are only computed and processed by vehicles themselves (record as *local-pro-only*).
3) The data are only offloaded to RSU for computing and processing (record as *offload-only*).
4) The scheme similar to our proposal, except for collaborative computing when V2V action is adopted. That is to say, this scheme has no collaborative computing (record as *without collaborative computing*).

We evaluate the performance of average loss under different schemes and different data sizes. The data size indicates the amount of data generated in each time slot. The overall results are shown in Fig. 5. Overall, the average loss increases with the increasing data size. Specifically, the variation process of average loss can be divided into four phases. The first phase is when data size varies from about 0.2 to 0.6. The second phase is when data size varies from about 0.6 to 1.0. The third phase is when data size varies from about 1.0 to 4.4. The fourth phase is when data size varies from 4.4 to 5.0. In the following, the four phases will be discussed in detail, respectively.

The first phase is shown as the circled part in Fig. 5(a). To show the variation process clearly, this part is zoomed in, as shown in the left figure of Fig. 5(b). In this phase, the average losses of *local-pro-only*, *our proposal*, and *without collaborative computing* have the same value and variation trending. However, the average loss of *offload-only* is much higher than the other three schemes. This is because the local computing capacity of vehicles is sufficient to process the data when data size is small, hence, there will be no penalty caused by unprocessed data. Since the cost of local computing is much lower than the cost of RSU computing (consisted of V2I

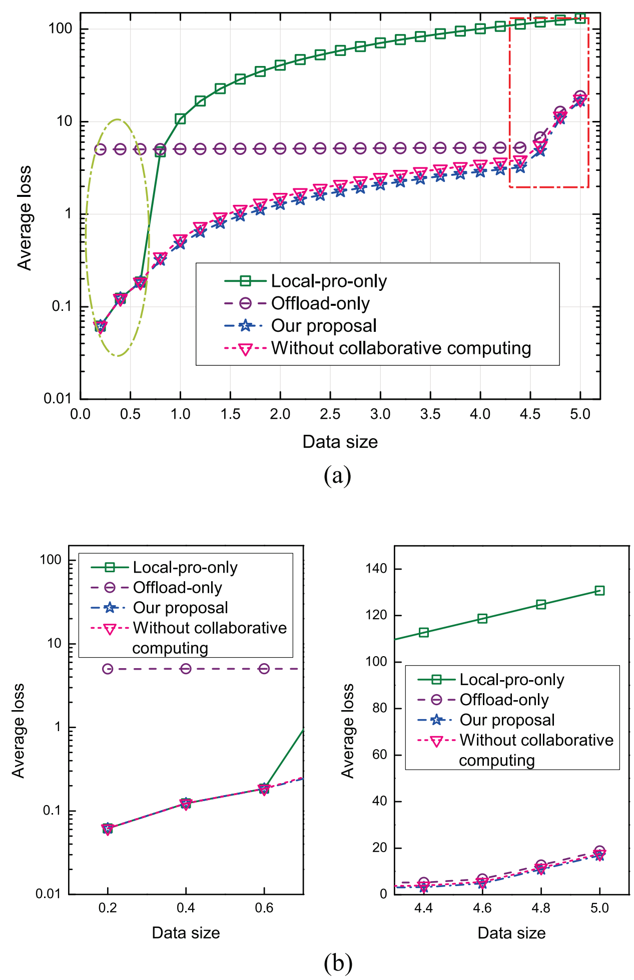

Fig. 5. Average loss with various data size and different schemes. (a) Data size varied from 0.2 to 5.0. (b) Circled part and framed part in Fig. 5(a).

communication cost and RSU computing cost), both agents of *our proposal* and *without collaborative computing* learn to adopt local computing actions for the purpose of minimizing the cost thus minimizing the loss. The underlying reason why the average loss of *offload-only* increases slowly is that the cost of V2I communication and RSU computing is relatively high.

TABLE IV
AVERAGE LOSS OF *Offload-Only* UNDER SMALL DATA SIZES

Data size	0.2	0.4	0.6	0.8
Average loss	5.0051	5.0158	5.0268	5.0379

In each time slot, the data of vehicles are offloaded to RSU through V2I communication, and RSU computes the offloaded data. The communication and computation costs are relatively fixed because all the costs are calculated in the unit of time slot. The cost of computing energy consumption of RSU is much lower than the communication and computation costs. Accordingly, the increase of data size would result in a small increase in loss caused by the increase of computing energy consumption. Another reason why the average loss looks like it increases slowly is that the vertical scale is of the exponential type. We tabulate the variation of average loss of *offload-only* in Table IV.

For the second phase, as the data size increases, the average loss of *local-pro-only* increases rapidly and exceeds that of *offload-only*. The underlying reason is that the data size in this phase exceeds the local computing capacity of vehicles, the task not meeting the delay constraints will result in a high loss because of the penalty to the system, as described in (10) and (11). It is noteworthy that the average losses of both *our proposal* and *without collaborative computing* are lower than that of *local-pro-only* and *offload-only*. This is because the former two schemes can optimally decide whether data are processed locally or offloaded to RSU, hence the data that exceed the local computing capacity are offloaded to RSU, for the purpose of minimizing the loss. In addition, the average loss of *our proposal* is lower than that of *without collaborative computing*. The underlying reason is that, in *our proposal*, the data can be migrated to collaborative vehicles through V2V communication for a lower cost thus a lower loss than being offloaded to RSU.

As data size keeps increasing, the third phase depicts the increase of average loss for all the four schemes. In this phase, the data size does not exceed the sum of local computing capacity and the data transmission capacity. In *our proposal* and *without collaborative computing*, the data can be partly offloaded to RSU for processing. Accordingly, no penalty is introduced, which has a lower average loss than *local-pro-only* and *offload-only*.

The fourth phase is shown as the framed part in Fig. 5(a). In this phase, the average losses of *offload-only*, *our proposal*, and *without collaborative computing* have a rapid increase when the data size increases. The underlying reason is that, during this phase, the data size exceeds the sum of local computing capacity and the data transmission capacity. In this case, the penalty will be introduced due to unprocessed data, which results in a high loss according to (11). To show the variation process clearly, this part is also zoomed in, and the y-axis is changed to the linear scale, as shown in the right figure of Fig. 5(b). When the data size in this phase reaches a threshold (approximately 4.6), the average losses of *offload-only*, *our proposal*, and *without collaborative computing* have the same increasing trending as *local-pro-only*. This is because all the four schemes cannot completely process all the generated data when the data size exceeds a threshold. Accordingly, how fast the average loss increases depends on how fast the data size increases. Furthermore, we can find that the average loss of *our proposal* has almost the same values as that of the scheme without collaborative computing. This is because if the data are migrated to collaborative vehicles, a high loss will be caused due to the limited computing capacity of collaborative vehicles. Instead, during the training process, the agent learns that the data should be offloaded to RSU for processing to minimize the loss.

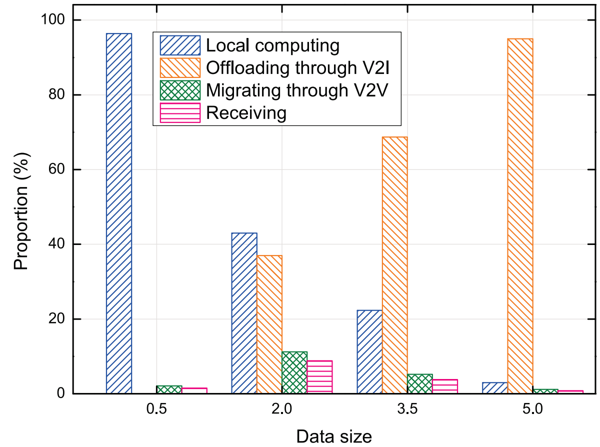

Fig. 6. Average proportion of data scheduling actions on the vehicle side under various data size.

3) Data Scheduling Actions: To understand how the agent of our proposal selects the optimal data scheduling actions during the training process, we evaluate the proportion of scheduling actions on both the vehicle side and RSU side. As comparison, we select four different data sizes, i.e., 0.5, 2.0, 3.5, and 5.0.

Fig. 6 shows the proportion of data scheduling actions on vehicle side under different data sizes. It is not difficult to find out that the action is gradually changing from local computing to offloading through V2I as data size increases. The reason is that when data size is small, the local computing capacity of vehicles is sufficient to process the data, the agent prefers to select local computing action for vehicles to minimize the cost thus minimize the loss. As the data size exceeds the local computing capacity of vehicles, the proportions of offloading through V2I, migrating through V2V, and receiving actions increase. This is because the task date should be processed under delay constraints to reduce the timeout penalty thus minimize the loss. The proportion of migrating through V2V action has almost the same value as the receiving action. The reason is that the establishment of a communication link must contain a sender and a receiver. It is noteworthy that the proportion of migrating through V2V action increases first then decreases as data size increases. The underlying reason is that, when data size is small, local computing is sufficient and the cost of migrating through V2V is higher than local computing. As data size increases and exceeds the local computing capacity, instead of offloading to RSU, data can be migrated to collaborative vehicles for processing because the cost of migrating through V2V is lower than offloading through V2I.

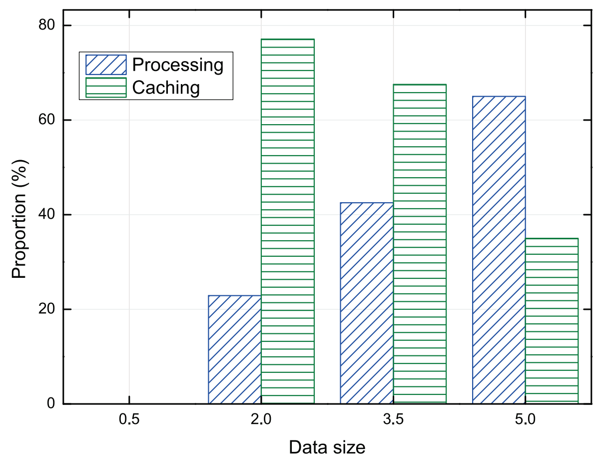

Fig. 7. Average proportion of data scheduling actions on the RSU side under various data size.

As data size keeps increasing, the proportion of migrating through V2V decreases until its value reaches almost 0. This is because if migrating through V2V action is adopted, the limited computing capacity of collaborative vehicles will cause much timeout penalty thus greatly increasing the loss, which violates the agent's purpose during the training process.

Fig. 7 shows the proportion of data scheduling actions on RSU side under different data sizes. There are two actions (i.e., processing data and caching data) related to RSU. It can be seen that the proportions of both processing and caching equal 0 when the data size is 0.5. The underlying reason is that under this value of data size, the local computing capacity of vehicles is sufficient to process data, the agent learns to choose local computing action during training to reduce cost thus minimize the loss. It is also consistent with the analysis for the first phase in Fig. 5. More data are offloaded to RSU as data size increases, the proportion of processing action also increases. This is because that there is a high probability that more data are cached in the queue with index 1 as the data size increases, the penalty would be thus introduced if data are not processed. In order to reduce the penalty thus minimize the loss, the proportion of processing data increases with the increase of data size. We can also draw a valuable conclusion that even when the data size is 5.0, the computing capacity of RSU is sufficient to process data. The underlying reason is that the proportion of processing action is not 100% when data size is 5.0, which indicates that the RSU processes data not in each time slot to reduce the cost of processing data thus minimize the loss.

VII. Conclusion

We argue that with the development of edge computing and autonomous driving, data communication and computation offloading will equally dominate the usage of bandwidth, and a unified framework to jointly consider all resources available in the network are necessary. As motivated, we have investigated the data scheduling problem in a unified paradigm where data can be processed locally, offloaded to RSUs, migrated to collaborative vehicles, or kept in caching queues. A multiqueue model for data caching on both the vehicle side and RSU side has been developed accordingly. To derive the optimal data scheduling strategy, we formulated an MDP model to reflect the interaction between data scheduling actions and loss, by jointly considering communication resource, caching state, computing resource, mobility of vehicles, as well as delay requirements of data. Next, we developed a DRL framework for the data scheduling problem, refining the state, action, and reward function, respectively. A DQN-based method with separate target Q-network was then proposed for solving this problem, which can efficiently optimize the data scheduling for minimizing the loss. Extensive simulation results were presented to illustrate that our proposal efficiently reduces data processing costs and helps data be processed under delay constraints.

In our future work, we intend to extend in three directions. First, we would include the migration of data between RSUs and consider the edge-cloud collaboration in vehicular networks. Second, we would apply the proposed algorithm in a real-world testbed and verify the performance. Third, since overhead would be generated when the state of each vehicle is collected, we would discuss and evaluate the overhead performance of the proposed method.

References

[1] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang. (2019). *Vehicular Edge Computing and Networking: A Survey*. [Online]. Available: https://arxiv.org/abs/1908.06849

[2] K. Zhang, S. Leng, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Artificial intelligence inspired transmission scheduling in cognitive vehicular communications and networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1987–1997, Apr. 2019.

[3] Y. Dai, D. Xu, S. Maharjan, G. Qiao, and Y. Zhang, "Artificial intelligence empowered edge computing and caching for Internet of Vehicles," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 12–18, Jun. 2019.

[4] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[5] C. Li, Q. Luo, G. Mao, M. Sheng, and J. Li, "Vehicle-mounted base station for connected and autonomous vehicles: Opportunities and challenges," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 30–36, Aug. 2019.

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

[7] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[8] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *Proc. IEEE Int. Conf. Commun. (IEEE ICC)*, 2017, pp. 1–6.

[9] P. Paymard, S. Rezvani, and N. Mokari, "Joint task scheduling and uplink/downlink radio resource allocation in PD-NOMA based mobile edge computing networks," *Phys. Commun.*, vol. 32, no. 2019, pp. 160–171, 2018.

[10] N. Cheng *et al.*, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[11] Q. Qi and Z. Ma. (2018). *Vehicular Edge Computing Via Deep Reinforcement Learning*. [Online]. Available: https://arxiv.org/abs/1901.04290

[12] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[13] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[14] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190–10203, Nov. 2018.

[15] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.

[16] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7635–7647, Oct. 2019.

[17] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[18] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[19] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1079–1092, Feb. 2019.

[20] Q. Luo, C. Li, T. H. Luan, and Y. Wen, "Optimal utility of vehicles in LTE-V scenario: An immune clone-based spectrum allocation approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 5, pp. 1942–1953, May 2018.

[21] M.-A. Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci, "Computation offloading game for an UAV network in mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (IEEE ICC)*, 2017, pp. 1–6.

[22] T. Q. Dinh, Q. D. La, T. Q. S. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6353–6367, Dec. 2018.

[23] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019.

[24] J. Li *et al.*, "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.

[25] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: on-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Oct. 2019.

[26] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 752–764, Jan. 2018.

[27] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2016, pp. 1–6.

[28] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[29] Y. Kim, J. Kwak, and S. Chong, "Dual-side optimization for cost-delay tradeoff in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1765–1781, Feb. 2018.

[30] O.-A. Maillard, T. A. Mann, and S. Mannor, "How hard is my MDP? The distribution-norm to the rescue," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1835–1843.

[31] V. Mnih, K. Kavukcuoglu, and D. Silver. (2013). *Playing Atari With Deep Reinforcement Learning*. [Online]. Available: https://arxiv.org/abs/1312.5602

[32] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[33] Didi Chuxing GAIA Initiative. *Urban Traffic Time Index and Trajectory Data (New)*. Accessed: Oct. 22, 2019. [Online]. Available: https://gaia.didichuxing.com

[34] M. Abadi *et al.* (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://arxiv.org/abs/1603.04467

Quyuan Luo received the B.E. degree from the Changchun University of Science and Technology, Changchun, China, in 2011. He is currently pursuing the Ph.D. degree in communication and information system with Xidian University, Xi'an, China.

He is also a Visiting Scholar of computer science with Wayne State University, Detroit, MI, USA, in 2019. His collaborator with Wayne State University is Prof. W. Shi. His current research interests include intelligent transportation systems, content distribution, edge computing, and resource allocation in vehicular networks.

Changle Li (Senior Member, IEEE) received the Ph.D. degree in communication and information system from Xidian University, Xi'an, China, in 2005.

He conducted his postdoctoral research in Canada and the National Institute of information and Communications Technology, Tokyo, Japan, respectively. He was a Visiting Scholar with the University of Technology Sydney, Ultimo, NSW, Australia. He is currently a Professor with the State Key Laboratory of Integrated Services Networks, Xidian University. His research interests include intelligent transportation systems, vehicular networks, mobile *ad hoc* networks, and wireless sensor networks.

Tom H. Luan (Senior Member, IEEE) received the B.E. degree from Xi'an Jiaotong University, Xi'an, China, in 2004, the M.Phil. degree from Hong Kong University of Science and Technology, Hong Kong, in 2007, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2012.

He was a Lecturer with the School of Information Technology, Deakin University, Melbourne, VIC, Australia, from 2013 to 2017. He is currently a Professor with the School of Cyber Engineering, Xidian University, Xi'an. His current research interests focus on the content distribution in vehicular networks, mobile cloud computing, and fog computing.

Weisong Shi (Fellow, IEEE) received the B.S. degree in computer engineering from Xidian University, Xi'an, China, in 1995, and the Ph.D. degree in computer engineering from the Chinese Academy of Sciences, Beijing, China, in 2000.

He is a Professor of computer science with Wayne State University, Detroit, MI, USA, where he is an Associate Dean for research of College of Engineering. His research interests include edge computing, big data systems, computer systems, energy efficiency computer systems, mobile and connected health, and connected and autonomous driving.

Prof. Shi received the Most Downloaded Paper Award in IEEE Computer; the National Outstanding Ph.D. Dissertation Award of China in 2002; the NSF CAREER Award in 2007; the Wayne State University Career Development Chair Award in 2009; the Charles H. Gershenson Distinguished Faculty Fellow in 2015; the College of Engineering Faculty Research Excellence Award in 2016; the Best Paper Award of ICWE'04, IEEE IPDPS'05, HPCChina'12, IEEE IISWC'12, and IEEE eHealth'17; the Best Paper Nominee Award of ACM UbiComp'14; and the Best Student Paper Award of IEEE HealthCom'15. He served as the Program/General Chair/Co-Chair for MetroCAD'19, MetroCAD'18, ACM/IEEE SEC'18, ACM/IEEE CHASE'18, ACM/IEEE CHASE'16, IEEE HotWeb'15, and IEEE CSE'11. He is a Distinguished Scientist of ACM. He is the Editor-in-Chief of *Smart Health* (Elsevier), an Associate Editor-in-Chief of IEEE INTERNET COMPUTING, an Associate Editor of the IEEE TRANSACTIONS ON SERVICES COMPUTING, the *ACM Transactions on Internet of Things*, and *Computing* (Springer), and the Editor of *IEEE Blockchain Newsletter*, IEEE INTERNET COMPUTING, *Sustainable Computing: Informatics and Systems*, the *Journal of Computer Science and Technology*, and the *International Journal of Sensor Networks*.