Deep Reinforcement Learning-Empowered Task Offloading for Efficient DNN Partition in Vehicular Edge Computing

Huaming Wu, *Senior Member, IEEE*, Fengyu Li, and Huijun Tang, *Member, IEEE*

***Abstract*—Deep neural networks (DNNs) have driven breakthroughs in autonomous driving through end-to-end methods, utilizing their powerful learning capabilities to generate vehicle controls directly from sensor data. However, maximizing the satisfaction of DNN inference requirements under the constraints of limited computing and energy resources on the vehicle side has emerged as a critical challenge in Vehicular Edge Computing(VEC). To address these challenges, we propose the reinforcement learning-empowered task diversion scheduling algorithm named RTD. This algorithm intelligently offloads computationally intensive portions of the DNN to Roadside Units (RSUs) by taking into account factors such as the battery coefficient and the type of DNNs. Firstly, we utilize the FLOPs method to model the data flow structure and computational load distribution of the DNNs. Subsequently, we formulate the task offloading model as an optimization problem that jointly considers latency, energy consumption, and the remaining battery power of the vehicle. Finally, after simplifying the optimization problem using the diversion algorithm, we employ the SAC method to determine the optimal offloading strategy. Extensive experiments demonstrate that RTD significantly reduces overall task completion time, effectively handles time-sensitive tasks, properly protects low-battery vehicles, and adapts well to dynamic network environments.**

***Index Terms*—DNN partition, Roadside units, Task offloading, Deep reinforcement learning**

I. Introduction

With the development of wireless communication technology and the continuous progress of technologies such as cloud computing and edge computing, the Internet of Vehicles (IoV) is constantly updated and evolved as a significant component of autonomous driving technology [1]. IoV centers on facilitating communication and connectivity between vehicles, enabling data transmission between vehicles and infrastructure to improve road safety, traffic efficiency, and overall transportation performance [2, 3]. However, as road environments become increasingly complex, a growing amount of time-sensitive tasks need to be processed to ensure the normal running of autonomous driving. Meanwhile, the ever-increasing demands of safety and convenience for users have further presented challenges to the development of autonomous driving [4, 5].

A promising approach to this challenge is Vehicular Edge Computing (VEC) [6–8], which offloads computational tasks from vehicles to roadside units (RSUs) deployed along the roadway. With advanced computational capacities, RSUs can effectively meet the growing computational demands of vehicles. The assistance of RSU not only effectively alleviates the computational burden on vehicles, but also significantly reduces data processing latency [9, 10]. Additionally, VEC enables the aggregation and preprocessing of edge data, thereby mitigating network congestion and enhancing data privacy [11, 12]. Given its numerous advantages, VEC has emerged as a cornerstone in the development of intelligent transportation systems, fostering more efficient, safer, and smarter vehicle environments [13–15].

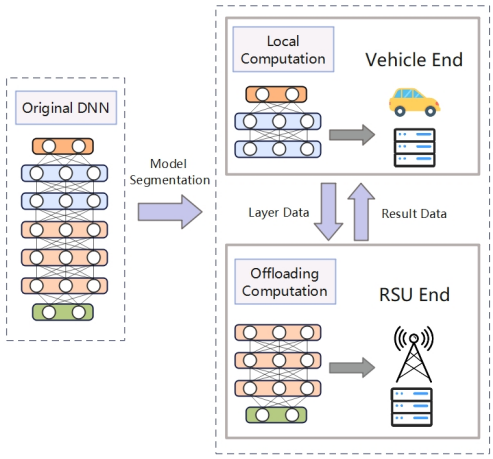

Fig. 1. Paradim of DNN partitioning in VEC networks

While VEC networks have unique advantages in reducing inference latency, they may have potential performance bottlenecks when facilitating distributed DNN inference [10, 12]. They usually have limited computing power, which makes it challenging to drive inferences based on the entire DNN model. Therefore, we adopt the DNN partitioning method to fully accelerate the inference process. DNN partitioning can decompose the entire DNN model into multiple blocks or segments and execute these components independently on different devices to facilitate distributed inference [14].

In this study, we consider implementing the task offloading algorithm on the granularity of splitting DNN layers, which can be shown as Fig.1. Firstly, a diversion algorithm is utilized to confirm which RSU the task will be offloaded to at the vehicle side. Then, an algorithm based on SAC is

H. Wu and F. Li are with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China. E-mail: {whming, fengyu_li}@tju.edu.cn
H. Tang is with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China. Email: tanghuijune@hdu.edu.cn.
(Corresponding author: Huijun Tang)

2025 IEEE International Conference on Web Services (ICWS) | 979-8-3315-5563-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICWS67624.2025.00055

2836-3868/25/$31.00 ©2025 IEEE
DOI 10.1109/ICWS67624.2025.00055

adopted to determine the optimal offloading decisions and resource allocation at a granularity of DNN layers. The main contributions of this paper are as follows:

- We utilized the FLOPs method to model the computational and transmission dataflow of DNNs. Compared with traditional experimental testing methods, this approach provides a more specific representation of dataflow structure and distribution.
- We designed a novel reinforcement learning-empowered task diversion algorithm to allocate part of the task offloading decision-making power to the vehicle side. The vehicle side will determine the target RSU for offloading based on the information broadcast by the RSUs, enabling the diversion of tasks from the vehicle side.
- We consider two types of parallelism during the task execution process. The first type is that the local computation and task transmission processes are parallelized with the queuing process at the RSU end. The second type of parallelism occurs when a new task can be transmitted to the RSU during the execution of the preceding task.
- We take into account the remaining battery power of the vehicles using the battery factor α. We developed customized task offloading strategies based on the current battery status of individual vehicles. For vehicles with low battery levels, the algorithm increases the weight of local energy consumption.

The remainder of this paper is organized as follows: Section II presents the modeling of the task offloading process and DNN partitioning, formulating the entire problem as an optimization expression. Section III introduces the diversion algorithm and provides the simplified model formulation after applying diversion algorithm. Subsequently, we detail the SAC-based task offloading algorithm. Finally, Section IV presents evaluations of the proposed algorithms.

II. System Model and Problem Formulation

A. Vehicular Network Scenarios

The scenario depicted in Fig. 2 involves a straight two-way urban road where RSUs, capable of handling offloading tasks, are placed at regular intervals. The communication model between RSUs and vehicles is based on vehicle-to-infrastructure (V2I) communication [16]. Vehicles collaborate with RSUs to decide whether to perform local computation or partial offloading. In recent years, the rapid advancement of deep learning and its groundbreaking applications in autonomous driving and visual recognition have prompted this paper to focus on a partial task-offloading scheme for neural networks.

B. DNN Partitioning Model

1) Basic Settings: Vehicles are designated as $veh = \{1, 2, \cdots, v, \cdots, V\}$. The division of time slot intervals continues to be utilized, with the time interval being defined as $\tau = \{1, 2, \cdots, T\}$. The generated task can be defined as:

$$\varphi_m^v(t) = \{d_m(t), DDL_m(t), x_m, y_m\}, \tag{1}$$

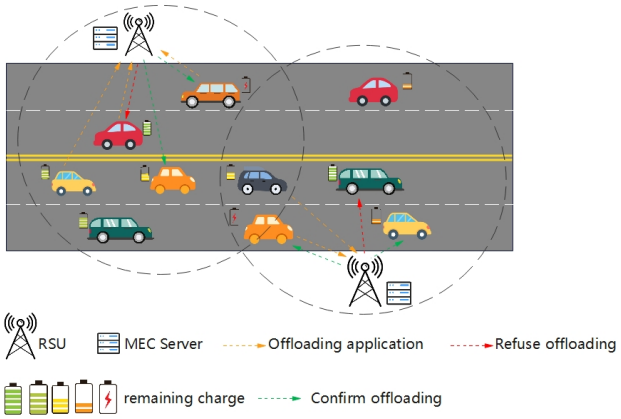

Fig. 2. Topology of vehicular network model

where $\varphi_m^v(t)$ represents the task m generated by vehicle v at time t, and $d_m(t)$ denotes the type of DNN task m at time t. $DDL_m(t)$ represents the time constraint for this task. Failure to complete the task within the designated time limit will result in a significant penalty. x_m represents the offloading variable of the task, and y_m denotes the index of the RSU to which the task is offloaded.

2) DNN Offloading Process: In this study, we consider a set of multiple DNNs, denoted as $G = \{G_1, \cdots G_i, \cdots, G_I\}$. For each DNN G_i, the layers are defined as follows:

$$G_i = \left\{l_i^1, \cdots l_i^j, \cdots l_i^J\right\}. \tag{2}$$

During the offloading process, the subsequent part of a specific layer can be offloaded to the RSU for computation. Therefore, we define the offloading decision variable as $x_m(t) \in \{0, 1, 2, \cdots, J\}$, which determines how many layers of the DNN will be offloaded to the RSU. For instance, if our offloading decision variable is indicated as $x_i = j$, then we will perform the computations of layers from l_i^1 to l_i^j locally and offload the layers after l_i^{j+1} to the RSU. If our current decision variable is indicated as $x_i = J$, this task requires local computation by the vehicle.

Considering the granularity of DNN layers, it is essential to model the floating-point operations (FLOPs) [17] and data traffic for each layer of the neural network defined as:

$$l_i^j = \left\{C_i^j, D_i^j\right\}, \tag{3}$$

where C_i^j represents the computation cost and D_i^j represents the data traffic. Subsequently, we will employ the FLOPs method to calculate C_i^j, D_i^j of the fully connected layers, pooling layers, and convolutional layers.

Regarding the convolutional layer and the pooling layer, the corresponding FLOPs and data traffic are defined as follows:

$$C_i^j = C_{\text{in}} \left(k^2 + k^2 - 1\right) \times H \times W \times C_{\text{out}}, \tag{4}$$

$$D_i^j = C_{\text{in}} \times H^{j-1} \times W^{j-1}, \tag{5}$$

where C_{in} and C_{out} are the input channel and output channel of the current layer, k is the size of the kernel, H and W are the height and width of layer l_i^j, H^{j-1} and W^{j-1} are the

height and width of layer l_i^{j-1}.

As for the fully connected layer, its corresponding FLOPs and data traffic are:

$$C_i^j = (N_{\text{in}} + N_{\text{in}} - 1) \times N_{\text{out}}\,, \tag{6}$$

$$D_i^j = N_{\text{in}}, \tag{7}$$

where N_{in} is the number of neuron for the previous layer and N_{out} is the number of neuron for the next layer. $N_{\text{in}} + N_{\text{in}} - 1$ denotes the maximum number of operations required for a neural network layer to generate an output where up to N_{in} multiplications and $N_{\text{in}} - 1$ additions are needed.

C. Local and Offloading Computation Model

This section introduces the modeling for local task computing, RSU computing, and task transmission, primarily involving partial offloading, so we first derive the expressions for the FLOPs on both the vehicle and RSU sides. When the decision variable is set as x_m, the FLOPs for each part is as:

$$C_{veh} = \sum_{j=0}^{x_m-1} C_i^j,\ C_{RSU} = \sum_{j=x_m}^{J} C_i^j. \tag{8}$$

1) Local computing: In this case, we consider $x_m(t) = J$ for DNN with a total layer of J. The completion time is only related to the execution time, given by:

$$t_m^{veh}(t) = \frac{C_{veh}}{f_{v,m}^{veh}(t)} = \frac{\sum_{j=0}^{x_m(t)-1} C_i^j}{f_{v,m}^{veh}(t)}, \tag{9}$$

where $f_m^{veh}(t)$ is computation capacities allocated to task m by the current vehicle, C_i^j is defined in Eq. (4). The energy consumed by the vehicle is:

$$e_m^{veh}(t) = \kappa \cdot C_{veh} \cdot \left(f_{v,m}^{veh}(t)\right)^2, \tag{10}$$

where κ is the computation energy efficiency coefficient.

2) RSU computing: When parts of the DNN are offloaded to the RSU, the vehicle has to transmit the related data to the RSU first. The transmission time can be defined as:

$$t_m^{trans} = \frac{D_i^j}{r_m(t)}, \tag{11}$$

where D_i^j can be calculated by Eq. (5) or Eq. (7), and $r_m(t)$ is the transmission rate of the task m at time t, which can be calculated by Shannon's formula [3], given by:

$$r_m(t) = B(t)\log_2(1 + \frac{P_m(t) \cdot \delta_{r,m}(t)}{N_0}), \tag{12}$$

where $B(t)$ is the bandwidth between the vehicle and RSU,$P_m(t)$ is the transmission power, $\delta_{r,m}(t)$ is the channel gain and N_0 is the Gaussian white noise. Similar to the local computing model, the processing time for RSU is defined as:

$$t_m^{RSU}(t) = \frac{C_{RSU}}{f_{r,m}^{RSU}(t)}, \tag{13}$$

where $f_{r,m}^{RSU}(t)$ is the computing capacities allocated to the offloading part of task m.

Meanwhile, the energy consumed by transmitting the task and processing the task at the RSU side is defined as:

$$e_m^{trans}(t) = \frac{D_i^j}{r_m(t)} \cdot P_m(t), \tag{14}$$

$$e_m^{RSU}(t) = \kappa \cdot C_{RSU} \cdot \left(f_{r,m}^{RSU}(t)\right)^2, \tag{15}$$

D. Problem Formulation

1) Task queuing in RSU: As shown in Fig. 2, it is common for multiple offloaded tasks to queue for the computation capacity. This study defines the task queue for RSU r as:

$$Q_r(t) = \{\varphi_{r,k}(t) \mid k = 1, 2, \cdots, K(t)\}, \tag{16}$$

where $\varphi_{r,k}(t)$ is the new symbol for task $\varphi_m^v(t)$ in the RSU queue, indicating that the task is positioned at the k-th place in the queue of RSU r.

So, we can define the queuing time for new task m as:

$$t_{r,m}^{queue} = \sum_{k=1}^{K(t)} t_{r,k}^{RSU}(t), \tag{17}$$

where $K(t)$ is the total queue length which may fluctuate with the time slot, $t_{r,k}^{RSU}(t)$ is the processing time for the k-th task, and the calculation formula is the same as Eq. (13).

2) Total time and energy consumption: When a queue is established, the total time of task m (same as the task queued $K(t)+1$ in RSU queue) can be defined as:

$$t_m = \max\left\{P_d + t_{r,m}^{queue}, P_g + t_m^{veh} + t_m^{trans}\right\} - P_g + t_m^{RSU}(t), \tag{18}$$

where P_d denotes the point of making the offloading decision. P_g represents the point of generating the current task.

Meanwhile, the total energy consumption can be defined as:

$$e_m = \alpha \cdot (e_m^{veh}(t) + e_m^{trans}(t)) + e_m^{RSU}, \tag{19}$$

where α represents a weight directly associated with the vehicle's current battery level $B_v(t)$. We can define the battery coefficient α as follows:

$$\alpha = \begin{cases} \frac{1}{B_v(t)}, & \frac{1}{3} < B_v(t) < 1, \\ 3, & 0 < B_v(t) \le \frac{1}{3}, \end{cases} \tag{20}$$

3) The optimization problem: From what we have discussed above, we can formulate the optimization problem as:

$$\boldsymbol{\mathcal{P}_1}: \min_{x,y,f} \frac{1}{M} \sum_{m=1}^{M} (t_m + \beta e_m) \tag{21}$$

$$\text{s.t.} \quad x_m(t) \in \{0, 1, \cdots, J\}, G_i = \left\{l_i', \cdots l_i^J\right\}, \forall t \in \tau, \tag{21a}$$

$$y_m(t) \in \{0, 1, \cdots, R\}, \forall t \in \tau, \tag{21b}$$

$$0 \le \sum_{m=1}^{M} f_{v,m}^{veh}(t) \le f_{max}^{veh}, \forall t \in \tau, \tag{21c}$$

$$0 \le \sum_{m=1}^{M} f_{r,m}^{RSU}(t) \le f_{max}^{RSU}, \forall t \in \tau, \tag{21d}$$

where β is the balance coefficient of time and energy consumption, Eq. (21a) indicates that the offloading decision variable is in the set $\{0, 1, \cdots, J\}$ for a specific task requiring the i-th DNN, which comprises a total of J layers. Eq. (21b) defines the RSUs that tasks may be offloaded to. Eq. (21c) restricts the computational resource on the vehicle side and Eq. (21d) limits the computational resource on the RSUs.

III. Proposed Approach

In Section II, we formulated the entire system model as an optimization problem. Nevertheless, it is a multi-objective optimization problem with an enormous state space. If we adopt the reinforcement learning approach for solving it now, converging may be tough. Therefore, we initially employ the pre-diversion algorithm to decompose the optimization problem into smaller subproblems. Subsequently, we utilize reinforcement learning to determine the optimal solution for each subproblem. An overview of the reinforcement learning-empowered task diversion scheduling can be shown in Fig. 3.

A. Pre-diversion DRL Algorithm

In this section, we elaborate on the 3 parts related to diversion and the DRL part is introduced in the next section.

1) Broadcast RSU state index: We stipulate that the RSU broadcasts its status index at fixed intervals of a certain number of time slots. This index is defined as:

$$RI_r = (t_{r,m}^{wait}, \delta_{m,r}(t), l_r, B_v(t)), \tag{22}$$

where RI_r represents the state index of RSU r, $t_{r,m}^{wait}$ is the waiting time for queuing and can be calculated by Eq. (17), l_r is location coordinate of the RSU r, $\delta_{m,r}(t)$ is the channel gain of RSU r and the vehicle, $B_v(t)$ is the bandwidth.

2) Offloading application: For a certain vehicle v, upon receiving the state index from multiple RSUs, an expected waiting Time t_r^{pre} will first be predicted based on the RSU index. Here, we define the prediction waiting time as:

$$t_r^{pre} = \frac{\max_j\{D_i^j\}}{r_m(t)} + t_{r,m}^{wait}, \tag{23}$$

where D_i is the output data size for DNN i.

Then, the vehicle will first choose the RSU with the shortest prediction waiting time, either t_{r1}^{pre} or t_{r2}^{pre} and follow the principle that the longer the expected waiting time is, the lower the probability of sending the offloading application to that RSU will be. The probability is defined as follows:

$$p_{r1} = \frac{t_{r2}^{pre}}{t_{r1}^{pre} + t_{r2}^{pre}},\ p_{r2} = \frac{t_{r1}^{pre}}{t_{r1}^{pre} + t_{r2}^{pre}}. \tag{24}$$

After confirming the probability of sending the application, an offloading application containing $\{l_v(t), f_{max}^{veh}, \varphi_m^v(t)\}$ will be sent to the RSU.

3) Offloading decision making: After receiving the task request from the vehicle, the RSU employs the DRL method to identify the potentially optimal routing decision $x_m(t)$. The procedure structure above can be summarized in Alg. 1.

Algorithm 1 Reinforcement learning-empowered task diversion scheduling algorithm

Input: $B_v(t)$,$Q_r(t)$,$\delta_{m,r}(t)$,$\varphi_m(t)$,f_{max}^{veh},f_{max}^{RSU}.
Output: Offlaoding decision $x_m(t)$.

```
1: for each time slot do
2:     Get φ_m(t) for each RSU;
3:     Broadcast RSU state index;
4:     for each vehicle with new task do
5:         Receive 2 sets of RSU state index;
6:         Predict the possible waiting time;
7:         Get p_r1 and p_r2 and send the offloading request;
8:     end for
9:     for each RSU do
10:        Receive offloading attempts;
11:        Make the offloading decision by SAC agent;
12:        Send the request feedback;
13:    end for
14: end for
```

We use the diversion method described by lines 2 to 9 in Alg. 1, where the algorithm delegates part of the offloading decision-making authority from RSUs to vehicles. Hence, we can transform the problem $\boldsymbol{\mathcal{P}_1}$ into problem $\boldsymbol{\mathcal{P}_2}$:

$$\boldsymbol{\mathcal{P}_2}: \min_{x,f} \quad t_m + \beta e_m \tag{25}$$

$$\text{s.t.} \quad x_m(t) \in \{0, 1, \cdots, J\}, G_i = \left\{1, \cdots l_i^J\right\}, \forall t \in \tau \tag{25a}$$

$$0 \le f_{v,m}^{veh}(t) \le f_{max}^{veh}, \forall t \in \tau \tag{25b}$$

$$0 \le f_{r,m}^{RSU}(t) \le f_{max}^{RSU}, \forall t \in \tau, \tag{25c}$$

B. SAC Based Offloading Decision Making

1) MDP modeling: We will model the problem as a Markov Decision Process (MDP) and present the state space, action space, and reward.

- **State**: Let $s(t)$ be the state space at time slot t. Hence, we can define the state as:

$$S(t) = [l_m(t), v_m(t), \phi_m^v(t), f_{max}^{veh}, f_{max}^{RSU}, Q_r(t)], \tag{26}$$

  where $l_m(t)$ is the location of the vehicle that generated the task m, $v_m(t)$ is the velocity of the vehicle, and $\phi_m^v(t)$ is the attribute of the task m. f_{max}^{veh} and f_{max}^{RSU} are the maximum computational resources of the vehicle and the current RSU.
- **Action**: Since we have eliminated the variable $y(t)$ in the optimization problem, according to problem P2,we can define the action as:

$$A(t) = [x_m(t), f_{v,m}^{veh}(t), f_{v,m}^{RSU}(t)], \tag{27}$$

  where $x_m(t)$ is the offloading decision variable of task m, $f_{v,m}^{veh}(t)$ and $f_{v,m}^{RSU}(t)$ are the computational resource from vehicle and RSU that can be allocated to task m, respectively.
- **Reward**: The reward function of our MDP can be defined as the negative of a weighted sum of delay and energy

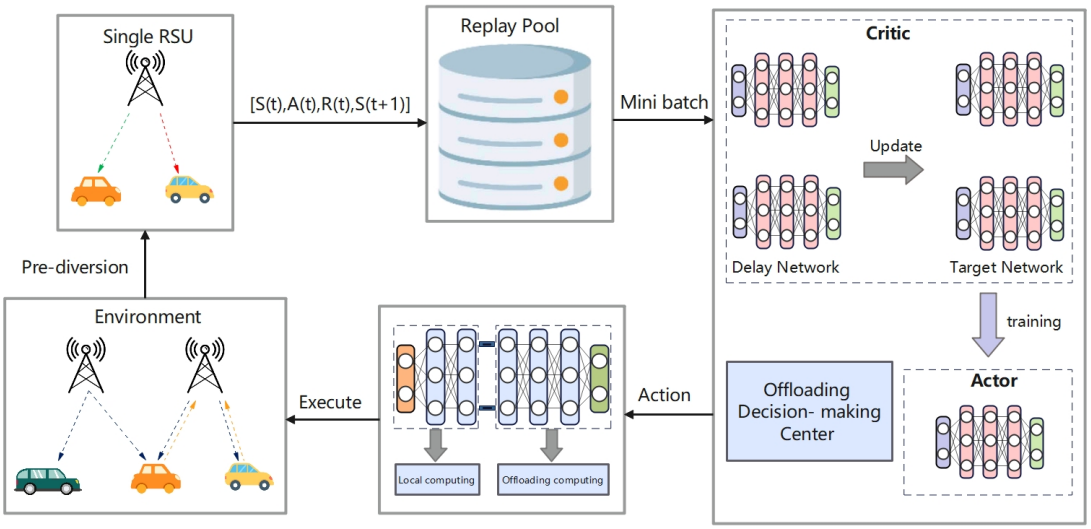

Fig. 3. Overview of the proposed reinforcement learning-empowered task diversion scheduling in VEC networks

consumption by:

$$R(t) = -(t_m + \beta e_m). \tag{28}$$

2) SAC algorithm: Unlike traditional reinforcement learning methods that merely maximize the reward, the SAC algorithm incorporates a weighted entropy term. This encourages the agent to explore the environment more thoroughly, prevents the system from prematurely converging to a suboptimal solution, and enhances the overall robustness of the system [18]. If $p(x)$ is the density function of a random variable X, we can define the entropy of X as:

$$H(X) = \mathbb{E}_{x\sim p}[-\log\left(p(x)\right)]. \tag{29}$$

In this algorithm, we modify the original reward to be a weighted sum of the cumulative reward and the entropy. As a result, we use this modified reward to derive the optimal policy $\pi^{\star}$. The following formula can express it:

$$\pi^{*} = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum r\left(s_t, a_t\right) + \mu H\left(\pi\left(\cdot \mid s_t\right)\right)\right], \tag{30}$$

where μ is the temperature parameter that can adjust the relative weights of the entropy and the reward to fine-tune the balance between exploration and exploitation [18].

As illustrated in Fig. 3, we utilize four critic networks: two delay networks and two target networks. Additionally, there is an actor network that learns the policy π^{*} based on the target network. The actions generated by the actor network are returned to the environment, where we will get (s_t, a_t, r_t, s_{t+1}) in the MDP model and store the transition quadruplet in the replay pool.

C. Computing Complexity

The complexity of Alg. 1 is $\mathcal{O}(TR)$ and depends on the total duration of the test intervals and the number of RSUs. Meanwhile, the complexity of SAC is $\mathcal{O}(ETN)$, which is mainly related to the number of training sessions, time intervals, and training rounds.

IV. Performance Evaluation

A. Parameter Settings

In the current scenario, the RSUs are elevated to 4 meters and furnished with a communication range of 500 meters [19]. In accordance with the standard of conventional highways, it supervises a three-lane expressway, with each lane 3.5 meters wide. In this study, we consider 4 types of DNN, including TinyTOLOv2, AlexNet, VGG16, and NiN.

The configuration is mainly derived from [20] and [21]. We set the channel bandwidth $B_v = 20$ MHz and the computation energy efficiency coefficient $\kappa = 10^{-26}$. We assume Gaussian white noise $N_0 = 3 \times 10^{-13}$. The maximum computational resources of the vehicle and RSU server are set as 3 GHz and 7 GHz. The range of temperature parameter μ is $[0.001, 1]$

B. Baselines

We implemented the following four benchmark algorithms for comparison with our proposed RTD scheme:

- **Local Executing Only (LEO)**: All tasks are handled entirely on the vehicle without being offloaded.
- **RSU Executing Only (REO)**: Regardless of the network condition, all tasks are directly offloaded to the RSUs.
- **Greedy-based Scheme (GBS)**: This method strictly adheres to the original greedy algorithm for task offloading without incorporating any additional enhancements [22].
- **Distributed DQN (DisDQN)**: This method uses the distributed DQN algorithm to partition DNN blocks and choose the offloading RSU [23].

C. Experiment Results

1) Convergence performance: As depicted in Fig. 4, a distinct convergence trend becomes evident after 2000 rounds.

Setting the learning rate to 0.05 can achieve the best convergence, and choosing the learning rate to 0.1 may achieve the highest reward with wider fluctuations. It is crucial to select an appropriate learning rate with care. Both large and small learning rates have their distinctive advantages. We set the learning rate as 0.05 for a trade-off between convergence and exploration.

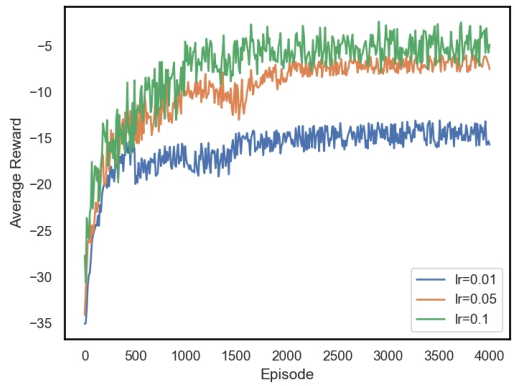

Fig. 4. Convergence of proposed method

2) Delay with changing number of vehicles: In Fig. 5, it can be observed that the performance disparities between the four baseline approaches and the proposed RTD under different vehicle numbers. When the number of vehicles increases, the latency of all approaches rises accordingly. With an increase in the number of vehicles, the number of tasks generated increases markedly, resulting in a corresponding increment in the total time needed to complete these tasks.

With the continuous growth of the number of vehicles, the waiting time for the baselines is increasing, and the increase in the waiting time is also gradually expanding. This delay variation is caused by task congestion. As for the RTD method, although the vehicle time delay is also on the rise, the increase in increment is not obvious.

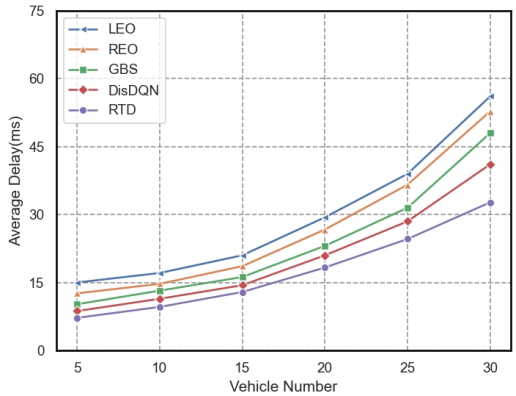

Fig. 5. Variation of delay with changing numbers of vehicles

3) Impact of task release rate for different algorithm: Fig. 6 illustrates the trend of the average total cost per task as the task release rate varies while maintaining a fixed number of 15 vehicles. In this scenario, the total cost of the LEO algorithm consistently remains the highest. This can be attributed to the penalty imposed on local computing when vehicle battery levels are low, as described in Eq. (19). Meanwhile, regardless of the task release rate, the proposed RTD algorithm consistently exhibits lower overall costs, demonstrating a clear advantage.

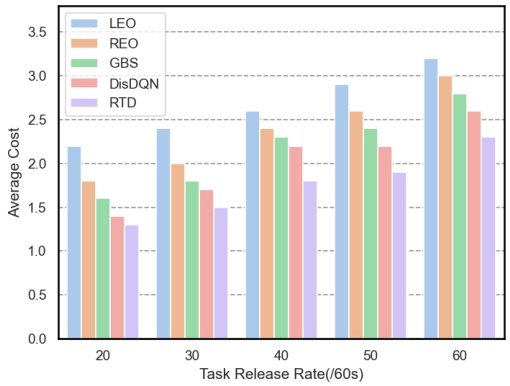

Fig. 6. Cost for algorithms with task release rate.

V. Conclusion

This paper proposes a joint computing offloading and resource allocation strategy in a VEC network to minimize the weighted sum of delay and energy consumption. Specifically, we address the remaining battery power of vehicles to provide customized and differentiated services for task offloading scenarios. Algorithmically, we implement pre-processing of tasks that require offloading, effectively reducing the action space for subsequent reinforcement learning decisions and thereby enhancing both the convergence and robustness of the reinforcement learning process. Simulation experiments demonstrate that the proposed method significantly improved the convergence of reinforcement learning. Additionally, it efficiently allocates computing and communication resources while prioritizing the normal running of low-battery vehicles, thereby reducing the overall system cost. However, despite its great simulation performance, the method may face challenges in practical implementation due to its complexity and the influence of environmental factors.

In future research, we plan to incorporate additional real-world traffic factors into our model to enhance its performance in complex and dynamic scenarios. Furthermore, we will integrate graph neural networks to further optimize decision-making and resource allocation, thereby improving the model's overall robustness and adaptability.

Acknowledgment

This work is supported by the Zhejiang Provincial Natural Science Foundation of China under Grant (No. LQN25F020016), the National Natural Science Foundation of China under Grant (No. 62401190 and 62071327), and the Tianjin Science and Technology Planning Project (No. 22ZYYYJC00020).

REFERENCES

[1] Y. Liang, H. Tang, H. Wu, Y. Wang, and P. Jiao, "Lyapunov-guided offloading optimization based on soft actor-critic for isac-aided internet of vehicles," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 14 708–14 721, 2024.

[2] M. Noor-A-Rahim, Z. Liu, H. Lee, M. O. Khyam, J. He, D. Pesch, K. Moessner, W. Saad, and H. V. Poor, "6g for vehicle-to-everything (v2x) communications: Enabling technologies, challenges, and opportunities," *Proceedings of the IEEE*, vol. 110, no. 6, pp. 712–734, 2022.

[3] Z. Liu, Z. Zhao, X. Wang, M. Dong, C. Qiu, and C. Zhang, "Toward mobility-aware edge inference via model partition and service migration," in *ICC 2023-IEEE International Conference on Communications*. IEEE, 2023, pp. 3258–3263.

[4] H. Wu, A. Gu, and Y. Liang, "Federated reinforcement learning-empowered task offloading for large models in vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 2, pp. 1979–1991, 2025.

[5] C. Tang, H. Wu, R. Li, and J. J. P. C. Rodrigues, "Joint optimization of task offloading content caching and resource allocation in vehicular edge computing," *ACM Trans. Auton. Adapt. Syst.*, Apr. 2025, just Accepted.

[6] C. Tang, Y. Zhao, and H. Wu, "Lyapunov-guided optimal service placement in vehicular edge computing," *China Communications*, vol. 20, no. 3, pp. 201–217, 2023.

[7] C. Li, L. Chai, K. Jiang, Y. Zhang, J. Liu, and S. Wan, "DNN partition and offloading strategy with improved particle swarm genetic algorithm in VEC," *IEEE Transactions on Intelligent Vehicles*, 2023.

[8] H. Tang, H. Wu, G. Qu, and R. Li, "Double deep q-network based dynamic framing offloading in vehicular edge computing," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1297–1310, 2023.

[9] R. Meneguette, R. De Grande, J. Ueyama, G. P. R. Filho, and E. Madeira, "Vehicular edge computing: Architecture, resource management, security, and challenges," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–46, 2021.

[10] Z. Hao, G. Xu, Y. Luo, H. Hu, J. An, and S. Mao, "Multi-agent collaborative inference via dnn decoupling: Intermediate feature compression and edge learning," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 6041–6055, 2022.

[11] X. Zhang, M. Peng, S. Yan, and Y. Sun, "Joint communication and computation resource allocation in fog-based vehicular networks," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13 195–13 208, 2022.

[12] Z. Liu, Q. Lan, and K. Huang, "Resource allocation for multiuser edge inference with batching and early exiting," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 1186–1200, 2023.

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

[14] M. Gao, R. Shen, L. Shi, W. Qi, J. Li, and Y. Li, "Task partitioning and offloading in dnn-task enabled mobile edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 4, pp. 2435–2445, 2021.

[15] C. Tang, G. Yan, H. Wu, and C. Zhu, "Computation offloading and resource allocation in failure-aware vehicular edge computing," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 1877–1888, 2024.

[16] M. S. Bute, P. Fan, G. Liu, F. Abbas, and Z. Ding, "A cluster-based cooperative computation offloading scheme for c-v2x networks," *Ad Hoc Networks*, vol. 132, p. 102862, 2022.

[17] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[18] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[19] C. Creß, Z. Bing, and A. C. Knoll, "Intelligent transportation systems using roadside infrastructure: A literature survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 7, pp. 6309–6327, 2024.

[20] S. Huang, M. Zhang, Y. Gao, and Z. Feng, "Mimo radar aided mmwave time-varying channel estimation in mu-mimo v2x communications," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7581–7594, 2021.

[21] H. Li, X. Li, Q. Fan, Q. He, X. Wang, and V. C. M. Leung, "Distributed dnn inference with fine-grained model partitioning in mobile edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 10, pp. 9060–9074, 2024.

[22] H. Gauttam, K. K. Pattanaik, S. Bhadauria, G. Nain, and P. B. Prakash, "An efficient dnn splitting scheme for edge-ai enabled smart manufacturing," *Journal of Industrial Information Integration*, vol. 34, p. 100481, 2023.

[23] C.-C. Lin, D.-J. Deng, Y.-L. Chih, and H.-T. Chiu, "Smart manufacturing scheduling with edge computing using multiclass deep q network," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4276–4284, 2019.