



Shi, Qiong, Cheng, Yongqiang and Li, Qingde (2026) Privacy-Aware Task Offloading in Vehicular Edge Computing Based on Deep Reinforcement Learning. In: 2026 31st International Conference on Automation and Computing (ICAC). IEEE, pp. 1-6. ISBN 979-8-3195-3407-1

Version: UNSPECIFIED

Available under license: Creative Commons: Attribution 4.0

Downloaded from: <https://sure.sunderland.ac.uk/id/eprint/20794/>

#### **Which version am I reading (articles, book chapters and books)**

Pre-print (or submitted version): the version of an article or chapter before any peer-review. Can be submitted to a publisher or added to a pre-print server.

AAM: Author Accepted Manuscript - The version of the article formally accepted for publication after the peer-review process. It includes the changes made by the author(s) in line with peer-reviewers' comments. It has not undergone the publisher's formatting, type-setting and copyediting.

VOR: Version of Record - The version published on the publisher's website which includes formatting, typesetting and copyediting.

#### **Usage guidelines**

Please refer to the usage guidelines at <http://sure.sunderland.ac.uk/policies.html> or alternatively contact [sure@sunderland.ac.uk](mailto:sure@sunderland.ac.uk).



# Privacy-Aware Task Offloading in Vehicular Edge Computing Based on Deep Reinforcement Learning

Qiong Shi  
School of Computer Science and  
Technology  
North University of China  
Taiyuan, China  
shiqiong0641@nuc.edu.cn

Yongqiang Cheng  
School of Computer Science  
University of Sunderland  
Sunderland, UK  
yongqiang.cheng@sunderland.ac.uk

Qingde Li  
School of Computer Science  
University of Hull  
Hull, UK  
q.li@hull.ac.uk

**Abstract**—Vehicular edge computing (VEC) faces critical challenges regarding limited onboard computational capacity and the potential for location privacy leakage. To address these issues, this paper proposes a novel, privacy-aware task offloading framework driven by deep reinforcement learning (DRL). First, we design a dynamic caching mechanism at roadside units (RSUs) to store previously computed task outputs, reducing redundant computation and communication overhead. Second, we introduce privacy entropy to quantitatively measure and enforce location privacy protection, mitigating the risk of RSU-induced data breaches. We then formulate a joint optimization problem that balances latency, energy consumption, and privacy preservation. To efficiently solve this, a deep deterministic policy gradient algorithm enhanced with prioritized experience replay (PER-DDPG) is developed to learn the optimal offloading strategy. Simulation results demonstrate that our proposed approach outperforms existing benchmark schemes, delivering superior convergence speed, lower latency, higher energy efficiency, higher privacy entropy and an improved task success rate.

**Keywords**—Task offloading, vehicular edge computing, deep reinforcement learning, privacy protection, Internet of Vehicles.

## I. INTRODUCTION

The rapid development of the Internet of Vehicles (IoV) has enabled emerging applications, such as autonomous driving and intelligent transportation systems, which require high computational capacity and low latency [1]. However, the limited computing capability of onboard units (OBUs) makes efficient task offloading essential [2]. Vehicular edge computing (VEC) provides the ideal infrastructure for this by deploying edge servers at roadside units (RSUs), allowing vehicles to seamlessly offload computations tasks to nearby nodes for faster, energy-efficient processing[3].

Existing studies on task offloading in IoV primarily focus on optimizing latency and energy consumption. For latency optimization, Tang et al [4] proposed a distributed dynamic offloading algorithm to address the uncertainty of edge node workloads, thereby reducing the average task delay. For energy optimization, Zhou et al [5] developed a double deep Q-learning-based approach to minimize long-term system energy consumption under uncertain task demands.

For joint optimization of latency and energy consumption, Chen et al [6] combined game theory with stochastic optimization to enable multi-edge server selection, Zhou et al [7] alleviated resource bottlenecks by constructing a multi-vehicle cooperative offloading framework, and Zhao et al [8] adopted deep reinforcement learning (DRL) to perform fine-grained task partitioning, thereby reducing overall computational overhead. Although these studies have achieved notable improvements in system performance, they largely overlook privacy leakage issues during the task offloading process.

In mobile edge computing environments, attackers can infer vehicle locations by analyzing task offloading behaviors, which poses significant security risks[9]. Existing studies have therefore focused on location privacy protection. Specifically, He et al [10] reduced location inference risks by optimizing offloading strategies, He et al [11] introduced fake tasks to interfere with attackers' inference processes, and Wang et al [12] achieved a trade-off between privacy preservation and computational performance using differential privacy techniques. However, these approaches typically incur considerable computational or communication overhead, limiting their applicability in latency-sensitive IoV scenarios.

With the increasing dynamics of IoV environments, DRL has emerged as an effective approach for solving task offloading problems. In particular, Uddin et al [13] employed a Deep Q-Network(DQN)-based method to enable priority-aware task offloading under latency constraints, Li et al [14] utilized an improved Proximal Policy Optimization (PPO) algorithm for resource scheduling to reduce latency and enhance energy efficiency, Zhao et al [15] integrated Lyapunov optimization with the SAC algorithm to achieve joint optimization of latency and system stability, and Qiu et al [16] adopted the Deep Deterministic Policy Gradient (DDPG) algorithm to handle continuous action spaces, enabling comprehensive optimization of latency and energy consumption. Despite the promising performance of DRL-based methods in complex environments, they still suffer from several limitations, including high training cost, slow convergence, and difficulties in achieving effective multi-objective optimization.

To overcome these limitations, we propose a privacy-aware DRL-based task offloading scheme. The main contributions of this paper are summarized as follows:

- (1) **Dynamic Caching for Efficiency:** We design a popularity-based dynamic caching mechanism to minimize redundant computations and reduce transmission overhead.
- (2) **Quantitative Privacy Protection:** We introduce privacy entropy as a rigorous metric to quantitatively evaluate and safeguard location privacy during the offloading process.
- (3) **Joint Optimization via PER-DDPG:** We formulate a joint optimization framework that balances latency, energy consumption, and privacy. To solve this, we develop a PER-DDPG algorithm, which demonstrably improves convergence efficiency and the quality of offloading decisions.

## II. SYSTEM MODEL

### A. Network Model

As shown in Figure 1, we consider a VEC system deployed over a square area of size  $L \times L$ , consisting of  $N$  RSUs and  $M$  vehicles. Each RSU is equipped with an edge server characterized by computing capacity  $C_n$  and storage capacity  $S_n$ . While vehicles are capable of performing local computation, they may offload tasks to RSUs when necessary to meet performance demands. The base station (BS) acts as the agent in the reinforcement learning framework, responsible for perceiving the system state and making offloading decisions. The system operates over discrete time slots. In a given time slot  $T_t$ , vehicle  $V_m$  generates the  $i$ -th computational task, denoted as  $D_{m,i} = \{d_{m,i}, c_{m,i}, T_{m,i}^{max}\}$ , where  $d_{m,i}$  denotes data size,  $c_{m,i}$  represents required CPU cycles, and  $T_{m,i}^{max}$  is the maximum tolerable latency. Because each task can be processed either locally or offloaded to an RSU, we define the offloading decision variable  $a_{m,i}$  as follows: local execution occurs if  $a_{m,i} \leq 0$ , whereas offloading to an RSU occurs if  $a_{m,i} > 0$ .

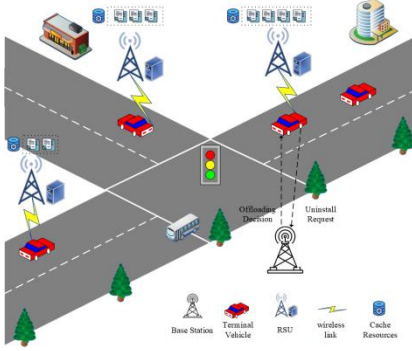


Fig. 1. VEC Task Offloading Network Architecture

### B. Communication Model

The uplink transmission rate between a vehicle and an RSU is given by:

$$r_{D_{m,i}} = B \log_2 \left( 1 + \frac{P_{D_{m,i}}^{tran} \times h_{V_m, R_n}}{\sigma^2} \right) \quad (1)$$

Where,  $B$  denotes the uplink bandwidth of the vehicle;  $\sigma^2$  represents the noise power of the wireless channel; and  $P_{D_{m,i}}^{tran}$  is the transmission power of the task.

In (1),  $h_{V_m, R_n}$  represents the channel gain between vehicle  $V_m$  and  $R_n$ , which can be expressed as:

$$h_{V_m, R_n} = g \left( \frac{l_0}{l_{V_m, R_n}} \right)^\xi \quad (2)$$

where  $g$  denotes the path loss constant,  $l_0$  is the reference distance,  $l_{V_m, R_n}$  represents the distance between  $V_m$  and  $R_n$ , and  $\xi$  denotes the path loss exponent (Let  $\xi=2$ ) [17].

### C. Caching Model

To enhance system efficiency and minimize redundant processing, we introduce a dynamic caching mechanism. Under this model, RSUs store popular task data and their corresponding computation results. However, because RSUs possess strictly limited storage capacity, caching every offloaded task is infeasible. Consequently, the caching

decisions at each RSU must adhere to the following storage constraint:

$$\sum_{i=1}^I s_n^i \leq S_n \quad (3)$$

where  $I$  represents the total number of cached tasks,  $s_n^i$  denotes the storage requirement of task  $i$ , and  $S_n$  is the maximum storage capacity of RSUs  $R_n$ .

### D. Computation Model

The total task completion time generally consists of both transmission latency and computation latency. Within our system, tasks generated by vehicles can be processed locally or offloaded to RSUs via V2R links. To systematically evaluate system performance, we analyze latency and energy consumption across three distinct scenarios: local execution, RSU offloading, and caching.

#### 1) Local Computation

When a task  $D_{m,i}$  is processed locally, its computation latency  $T_{m,i}^{local}$  and energy consumption  $E_{m,i}^{local}$  are respectively given by:

$$T_{D_{m,i}}^{local} = \frac{c_{m,i}}{f_m^{local}} \quad (4)$$

$$E_{D_{m,i}}^{local} = \kappa (f_m^{local})^2 c_{m,i} \quad (5)$$

where  $c_{m,i}$  denotes the required CPU cycles to execute task  $D_{m,i}$ ,  $f_m^{local}$  is the local CPU frequency, and  $\kappa$  represents the energy coefficient of the vehicle's processor.

#### 2) RSU Computation

Alternatively, when task  $D_{m,i}$  is offloaded to an RSU, it incurs both transmission and computation costs. The transmission latency  $T_{D_{m,i}}^{tran}$  and energy consumption  $E_{D_{m,i}}^{tran}$  are given by:

$$T_{D_{m,i}}^{tran} = \frac{d_{m,i}}{r_{D_{m,i}}} \quad (6)$$

$$E_{D_{m,i}}^{tran} = P_{D_{m,i}}^{tran} \times T_{D_{m,i}}^{tran} \quad (7)$$

where  $d_{m,i}$  is the data size of the task  $D_{m,i}$ ,  $r_{D_{m,i}}$  is the transmission rate of  $D_{m,i}$ , and  $P_{D_{m,i}}^{tran}$  denotes the transmission power.

Once successfully transmitted, the subsequent computation latency  $T_{D_{m,i}}^{comp}$  and energy consumption  $E_{D_{m,i}}^{comp}$  at the RSU are defined as:

$$T_{D_{m,i}}^{comp} = \frac{c_{m,i}}{f_n^{RSU}} \quad (8)$$

$$E_{D_{m,i}}^{comp} = \delta \times T_{D_{m,i}}^{comp} \quad (9)$$

Where  $f_n^{RSU}$  denotes the CPU frequency of the RSU and  $\delta$  represents the computation power of the RSU server.

Consequently, the total latency and energy consumption for an offloaded task are:

$$T_{D_{m,i}}^{V2R} = T_{D_{m,i}}^{tran} + T_{D_{m,i}}^{comp} \quad (10)$$

$$E_{D_{m,i}}^{V2R} = E_{D_{m,i}}^{tran} + E_{D_{m,i}}^{comp} \quad (11)$$

### 3) Caching Computation

If the content of task  $D_{m,i}$  is already cached at the RSU, the vehicle simply retrieves the computed result without reoffloading the task. Because the result size is typically very small, the associated latency and energy consumption are considered negligible and approximated as zero:

$$T_{D_{m,i}} = 0 \quad (12)$$

$$E_{D_{m,i}} = 0 \quad (13)$$

## III. PRIVACY-AWARE PER-DDPG OFFLOADING METHOD

### A. Dynamic Caching Strategy

In our proposed system, the BS acts as a central agent that monitors the offloaded tasks received during each time slot. To optimize storage, the BS calculates the request frequency of each task type to evaluate its popularity, thereby classifying tasks as either popular or unpopular. By merging current requests with historical data, the system identifies dynamically popular tasks. Ultimately, based on this dynamic popularity sequence and the strictly limited storage capacity of the BS, specific tasks are selected for caching. These caching outcomes subsequently inform offloading decisions, dictating whether tasks should be executed locally or offloaded.

#### 1) Dynamic Popularity Computation

First, the RSU computes the popularity of each task content  $p_i$  by aggregating the number of service requests  $P_i^T$  over a specific time period  $T$ ,

$$P_i^T = \sum_{t=0}^T P_i^t \quad (14)$$

The accumulated requests,  $P_i^T$  are then sorted in descending order, and the average value  $\overline{P_i^T}$  is established as the threshold. Task contents that meet or exceed this threshold are designated as the historical popular task set at RSU  $R_n$ , i.e.,

$$P_{R_n}^{pre} = \{p_1, p_2, \dots, p_i\} \quad (15)$$

To enable dynamic load balancing and enhance overall resource utilization, each RSU periodically reports its load status to the BS which then allocates popular task contents to RSUs experiencing lower loads.

At the onset of each time slot, the historical set  $P_{R_n}^{pre}$  is updated by comparing it against current popular task contents and recalculating request frequencies, ultimately producing the finalized popular task content set  $P_{R_n}^t$ .

#### 2) 0-1 Knapsack-Based Caching Strategy

Given the heterogeneous sizes of tasks and the strictly limited storage capacity of RSUs, we employ a 0-1 knapsack-based caching strategy to store high-value tasks within resource constraints[18]. This approach aims to optimally select non-redundant combination of tasks—each with distinct values and storage requirements—to maximize the total cached value without exceeding the RSU's capacity. The task value is explicitly defined by its popularity, which corresponds to its request frequency in the current time slot. This 0-1 knapsack problem is resolved via dynamic programming, yielding the optimal set of cached task contents  $P_{R_n}^{cache}$ .

### B. Privacy Protection Model

In VEC networks, computationally intensive tasks are typically offloaded to the nearest RSUs with favorable channel conditions to minimize delay and energy consumption. However, by observing these offloading patterns, malicious attackers can infer vehicle locations. Such risks are exacerbated when multiple RSUs collude, enabling precise location. This threat is severely exacerbated if multiple RSUs collude, enabling precise trajectory inference and posing critical risks to driving safety and privacy.

To counteract this, an effective privacy-preserving mechanisms must strategically reduce the frequency of tasks sent to the nearest RSU and deliberately offload certain tasks to more distant RSUs, thereby confounding attackers. We introduce the concept of privacy entropy to quantitatively evaluate the privacy-preserving capability of the proposed offloading strategy[19]. Privacy entropy reflects the uncertainty an attacker faces: the higher the entropy, the lower the privacy leakage risk, and the better the protection of the vehicle's location privacy.

First, the task offloading preferences of a vehicle across different RSUs are described. Within time slot  $T_t$ , the total number of tasks offloaded by vehicle  $V_m$  is denoted as:

$$u_{V_m}^t = \sum_{n \in N} u_{V_m, R_n}^t \quad (16)$$

Here,  $u_{V_m, R_n}^t$  denotes the number of tasks offloaded by vehicle  $V_m$  to  $R_n$  during time slot  $T_t$ .

Next, the offloading preference of vehicle  $V_m$  for  $R_n$  can be expressed as the ratio of the number of tasks offloaded to  $R_n$  to the total number of tasks offloaded by  $V_m$ :

$$P_{V_m, R_n}^t = \frac{u_{V_m, R_n}^t}{u_{V_m}^t} \quad (17)$$

The information entropy  $H(X)$  of a discrete random variable  $X$  is defined as [20]:

$$H(X) = - \sum_{x \in X} p(x) \log p(x) \quad (18)$$

The corresponding privacy entropy of vehicle  $V_m$  can then be calculated based on its offloading preferences for the RSUs during time slot  $T_t$ , i.e.:

$$H_{V_m}^t = \begin{cases} - \sum_{n \in M} P_{V_m, R_n}^t \log_2 P_{V_m, R_n}^t & u_{V_m}^t > 0 \\ H_{\max} & u_{V_m}^t = 0 \end{cases} \quad (19)$$

When no tasks are offloaded, attackers cannot infer vehicle location, and privacy entropy is maximized. In contrast, offloading all tasks to the nearest RSU concentrates preferences, making location inference easier and minimizing privacy entropy.

Strategies ignoring privacy favor nearby RSUs to reduce delay and energy, which lowers privacy entropy. An effective strategy should balance privacy and performance by reducing reliance on nearby RSUs and increasing offloading to distant ones, enhancing privacy while keeping delay and energy within acceptable limits.

### C. Problem Formulation

To achieve both efficient and secure task offloading, we formulate a joint optimization problem designed to minimize delay and energy consumption while simultaneously maximizing privacy entropy

$$P: \max \Omega = \max (\omega_1 H_{V_m}^t - \omega_2 T_{V_m}^t - \omega_3 E_{V_m}^t) \quad (20)$$

subject to the following three primary constraints:

$$C1: u_{V_m, \text{local}} + u_{V_m, R_n} \leq u_{V_m} \quad (21)$$

$$C2: 0 < c_{m,i} < c_{R_n}^{\max} \quad (22)$$

$$C3: T_{m,i}^{\text{local}} \leq T_{m,i}^{\max}, T_{m,i}^{\text{V2R}} \leq T_{m,i}^{\max} \quad (23)$$

where,  $\omega_1$ ,  $\omega_2$ , and  $\omega_3$  are the weights of privacy entropy, delay, and energy consumption, respectively. Constraint C1 ensures that the total number of tasks processed locally and offloaded to RSUs does not exceed the total number of tasks generated by the vehicle. Constraint C2 ensures that the computational resources allocated to tasks do not exceed the maximum computing capacity of  $R_n$ . Constraint C3 ensures that the execution delay of each task, whether processed locally or at an RSU, does not exceed its maximum tolerable delay.

### D. PER-DDPG Algorithm

Because the formulated joint optimization problem is NP-hard, we model it as a Markov Decision Process (MDP) characterized by the tuple  $\{S, A, P, R\}$  and solve it utilizing a PER-DDPG.

1) *State Space*: The state space  $S_t$  at time  $t$  is defined as:

$$S_t = \{C_n, S_n, C_{V_m}^{\text{local}}, P_{R_n}^{\text{pre}}, P_{R_n}^t, P_{R_n}^{\text{cache}}, H_{V_m}, D_{m,i}\} \quad (24)$$

Here,  $C_{V_m}^{\text{local}}$  denotes the local computational resources of the vehicle;  $P_{R_n}^{\text{pre}}$ ,  $P_{R_n}^t$  and  $P_{R_n}^{\text{cache}}$  represent the historical popularity, current popularity, and cached task contents, respectively;  $H_{V_m}$  denotes the privacy entropy of the terminal vehicle and  $D_{m,i}$  represents the generated task by the vehicle.

2) *Action Space*: The agent determines the continuous offloading execution location variable  $a_{m,t}$  based on the state. Given the current state, the action space  $A_t$  at time  $t$ , can be defined as:

$$A_t = \{a_{m,i}\} \quad (25)$$

3) *Reward Function*: The reward function evaluates multiple performance metrics, including privacy protection, execution delay, and energy consumption, and can be expressed as:

$$R(S_t, A_t) = \sum_{m=1}^M (\beta_1 H_{V_m}^t - \beta_2 T_{V_m}^t - \beta_3 E_{V_m}^t) \quad (26)$$

Here,  $\beta_1$ ,  $\beta_2$ , and  $\beta_3$  denote the weights of the privacy reward, delay reward, and energy consumption reward, respectively.

To efficiently handle this continuous state-action space, the actor-critic-based DDPG learns optimal decisions by constructing a deterministic policy function that maximizes the cumulative system reward.

Further[21] more, the Prioritized Experience Replay (PER) mechanism enhances learning efficiency by prioritizing the replay experiences with larger temporal-difference (TD) errors. Because a larger absolute TD error indicates a more significant estimation discrepancy, prioritizing these experiences allows the agent to learn rapidly from its most substantial past mistakes, thereby improving overall policy convergence.

## IV. SIMULATION EXPERIMENTS AND RESULTS ANALYSIS

### A. Experimental Setup

The simulation platform was implemented using Python 3.8. The DDPG algorithm employs a four-layer deep neural network with the tanh activation function. In the simulation environment, 10 RSUs were deployed, each equipped with an MEC server. For vehicle tasks, the task size, required CPU cycles, and maximum tolerable delay were randomly selected within the ranges of 0.8–1.2Mb, 0.1–0.5G (Giga=10<sup>9</sup>), and 0.4–1.4s, respectively. The RSUs' computing capacities and storage were set to 1.4–2.4 GHz and 0–20 Mb, respectively, while vehicles' local computing capacity was set to 0.1–0.5GHz, with a wireless bandwidth of 10 MHz. The learning rates of the Actor and Critic networks were set to 0.0001 and 0.001, respectively. The replay buffer size was 6,400, the mini-batch size was 64, and the discount factor was set to 0.99.

For comparison, the Genetic Algorithm (GA), Deep Q Network (DQN), and Deep Deterministic Policy Gradient (DDPG) were implemented, with the only difference being the DRL algorithm used; all other experimental settings were kept identical. GA has strong global search capability and does not rely on gradient information, but suffers from slow convergence and high computational cost. DQN can handle high-dimensional state spaces and improves stability through experience replay, but is limited to discrete action spaces and may suffer from overestimation. DDPG is suitable for continuous action spaces and enables joint optimization of policy and value functions, but is sensitive to hyperparameters and may exhibit unstable training.

### B. Results Analysis

#### 1) Privacy Entropy under Different Task Arrival Rates

Figure 2 shows that task arrival rates have little impact on privacy entropy, which mainly depends on offloading preferences. The Local scheme achieves the highest privacy but suffers from high delay and energy consumption under

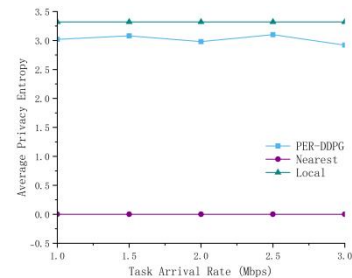


Fig. 2. Comparison of average privacy entropy under different task arrival rates

heavy load. The Nearest scheme has the lowest privacy entropy due to concentrated offloading, though it reduces delay and energy. PER-DDPG balances privacy and performance by optimizing delay, energy consumption, and

privacy entropy, enabling efficient multi-RSU offloading and improved overall performance.

### 2) Convergence and Overall Performance

Figures 3a–3d compare the cumulative reward, task success rate, system delay, and energy consumption of the four schemes over 1,000 episodes. The task success rate is defined as the ratio of the number of successfully completed tasks to the total number of tasks. As shown in Figures 3a and 3b, the PER-DDPG algorithm converges in approximately 200 episodes, achieving a cumulative reward of around -23 and a task success rate of 92.5%. Figures 3c and 3d show that, compared with DDPG, PER-DDPG reduces the system convergence delay by 1.4% and energy consumption by 6%.

PER-DDPG outperforms other methods in cumulative reward, task success rate, delay, and energy efficiency. This is mainly due to three factors: dynamic caching reduces computational overhead, the privacy entropy mechanism protects vehicle location information, and prioritized experience replay improves sampling efficiency, leading to better decisions and faster convergence.

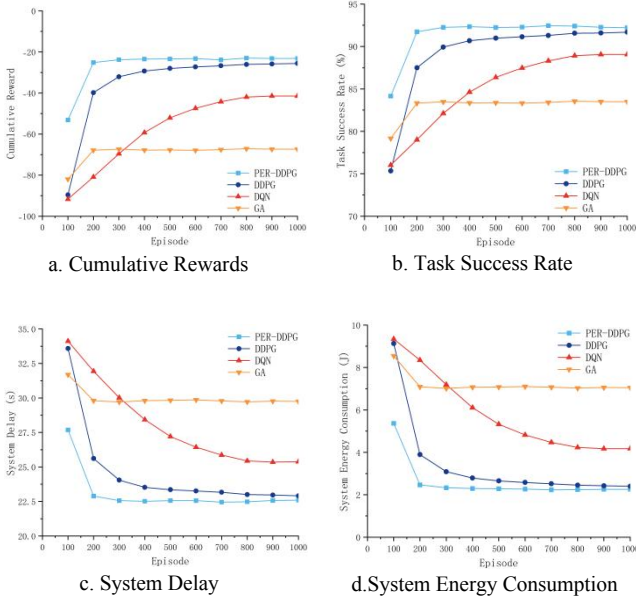


Fig. 3. Comparison of Algorithm Convergence Performance

### 3) Impact of Delay Constraints

Figure 4 shows task success rates under different delay constraints. As the maximum tolerable delay increases, all methods improve, since privacy entropy allows offloading to distant RSUs with higher latency but better privacy.

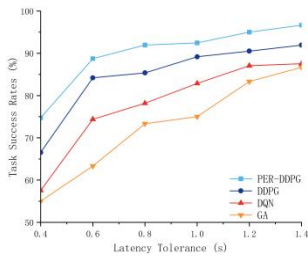


Fig. 4. Comparison of Task Success Rates under Different Delay Constraints

Under tight delay and limited resources, PER-DDPG maintains high success rates by jointly considering system state, privacy, and resource allocation. PER-DDPG improves the task success rate over DDPG, with gains ranging from 3.69% at the minimum to 12.35% at the maximum.

### 4) Impact of Edge and Vehicular Resource Variations

Figures 5 and 6 compare the delay and energy consumption of the four schemes under varying vehicle and RSU computing capacities. In Figure 5, PER-DDPG reduces delay by 4.4–12.8% and energy consumption by 25.5–45.9% compared to DDPG. In Figure 6, delay decreases by 2.5–10.3% and energy consumption by 10.7–37.5%.

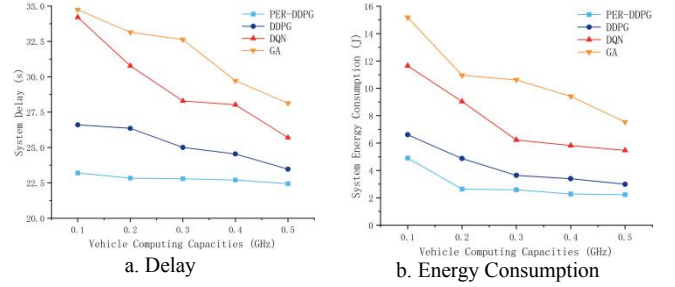


Fig. 5. Comparison of Delay and Energy Consumption under Different Vehicle Computing Capacities

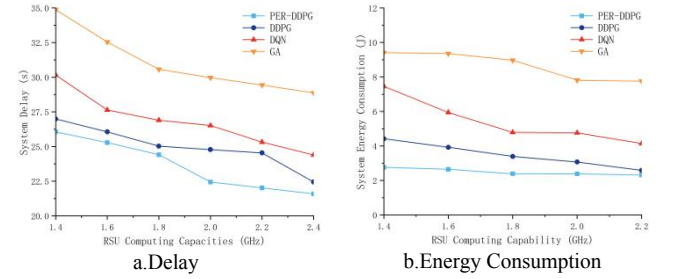


Fig. 6. Comparison of Delay and Energy Consumption under Different RSU Computing Capacities

Figure 7 compares the delay and energy consumption of the four schemes under different RSU cache capacities. As the cache size increases, both system delay and energy consumption decrease, since more completed tasks can be pre-cached and served without recomputation or transmission. PER-DDPG still delivers the best overall performance, reducing latency relative to DDPG by 1.73%–15.6% and energy consumption by 8.61%–64.5%.

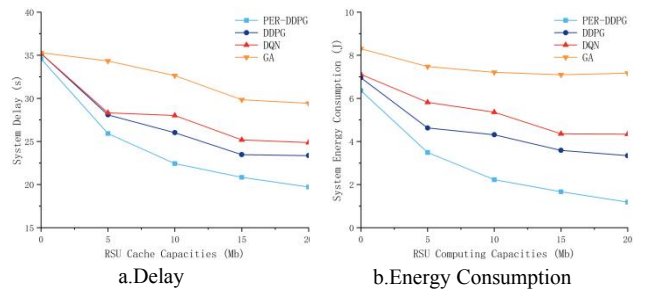


Fig. 7. Comparison of Delay and Energy Consumption under Different RSU Cache Capacities

Figure 8 compares delay and energy consumption for different task sizes. PER-DDPG reduces delay by 10.2–13% and energy consumption by 14–55.1% compared to DDPG.

### 5) Scalability Analysis

Figure 9 compares delay and energy consumption under different numbers of vehicles. PER-DDPG consistently achieves lower delay and energy consumption across all vehicle scales, PER-DDPG reduces latency relative to DDPG by 0.45%–12.7%, and lowers energy consumption by 14.3%–62.3%, while GA shows higher delay and energy use as the number of vehicles increases. This is because GA relies on offline or iterative search, making it less adaptive to the dynamic vehicular environment, whereas PER-DDPG can better handle environmental changes, demonstrating clear system performance advantages.

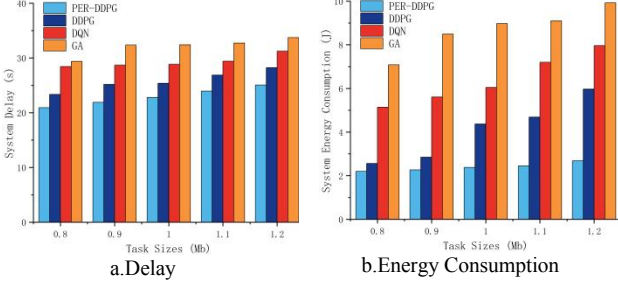


Fig. 8. Comparison of Delay and Energy Consumption under Different Task Sizes

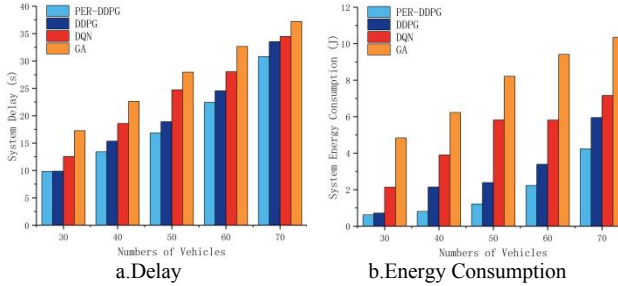


Fig. 9. Comparison of Delay and Energy Consumption under Different Vehicle Numbers

## V. CONCLUSIONS AND FUTURE WORK

To conclude, this work proposes a privacy-aware deep reinforcement learning scheme for vehicular task offloading. By integrating dynamic caching, privacy entropy, and a PER-enhanced DDPG algorithm, the proposed method improves offloading efficiency, load balancing, and training stability. Simulation results show that PER-DDPG converges within about 200 episodes, achieves up to 92.5% task success rate and higher privacy entropy, while reducing delay by up to 15.6% and energy consumption by up to 64.5% compared with DDPG.

Future work may incorporate vehicle trajectory prediction to reduce handover delays and improve dynamic offloading decisions, while edge server load prediction can further enhance overall performance.

## REFERENCES

- [1] Mishra P, Singh G. Internet of Vehicles for Sustainable Smart Cities: Opportunities, Issues, and Challenges[J]. *Smart Cities*, 2025, 8(3): 93.
- [2] Wang X, Lv J, Slowik A, Kim B-G, Parameshachari B D, Li K, Feng G. Augmented Intelligence of Things for Priority-Aware Task Offloading in Vehicular Edge Computing[J]. *IEEE Internet of Things Journal*, 2024, 11(22): 36002–36013.
- [3] Fan W, Su Y, Liu J, Li S, Huang W, Wu F, Liu Y. Joint Task Offloading and Resource Allocation for Vehicular Edge Computing Based on V2I and V2V Modes[J]. *IEEE Transactions on Intelligent Transportation Systems*, 2023, 24(4): 4277–4292.
- [4] Tang M, Wong V W S. Deep Reinforcement Learning for Task Offloading in Mobile Edge Computing Systems[J]. *IEEE Transactions on Mobile Computing*, 2022, 21(6): 1985–1997.
- [5] Zhou H, Jiang K, Liu X, Li X, Leung V C M. Deep Reinforcement Learning for Energy-Efficient Computation Offloading in Mobile-Edge Computing[J]. *IEEE Internet of Things Journal*, 2022, 9(2): 1517 – 1530.
- [6] Chen Y, Li K, Wu Y, Huang J, Zhao L. Energy Efficient Task Offloading and Resource Allocation in Air-Ground Integrated MEC Systems: A Distributed Online Approach[J]. *IEEE Transactions on Mobile Computing*, 2024, 23(8): 8129–8142.
- [7] Zhou Y, Chai Z-Y, Li Y-L, Li J-J. Parking Vehicle-Assisted Task Offloading in Edge Computing: A dynamic multi-objective evolutionary algorithm with multi-strategy fusion response[J]. *Swarm and Evolutionary Computation*, 2025, 94: 101900.
- [8] Zhao L, Zhao Z, Hawbani A, Liu Z, Tan Z, Yu K. Dynamic Caching Dependency-Aware Task Offloading in Mobile Edge Computing[J]. *IEEE Transactions on Computers*, 2025, 74(5): 1510–1523.
- [9] Li J, Ou W, Ouyang B, Ye S, Zeng L, Chen L, Chen X. Revisiting Location Privacy in MEC-Enabled Computation Offloading[J]. *IEEE Transactions on Information Forensics and Security*, 2025, 20: 4396–4407.
- [10] He Y, He X. Location Privacy-Aware Coded Offloading for Distributed Edge Computing[C]//IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).
- [11] Jing C, Xu J. PPDO: a privacy-preservation-aware delay optimization task-offloading algorithm for collaborative edge computing[J]. *Frontiers of Information Technology & Electronic Engineering*, 2025, 26(1): 27–41.
- [12] Wang Z, Sun Y, Liu D, Hu J, Pang X, Hu Y, Ren K. Location Privacy-Aware Task Offloading in Mobile Edge Computing[J]. *IEEE Transactions on Mobile Computing*, 2024, 23(3): 2269–2283.
- [13] Uddin A, Sakr A H, Zhang N. Adaptive Prioritization and Task Offloading in Vehicular Edge Computing Through Deep Reinforcement Learning[J]. *IEEE Transactions on Vehicular Technology*, 2025, 74(3): 5038–5052.
- [14] Li P, Xiao Z, Wang X, Huang K, Huang Y, Gao H. EPTask: Deep Reinforcement Learning Based Energy-Efficient and Priority-Aware Task Scheduling for Dynamic Vehicular Edge Computing[J]. *IEEE Transactions on Intelligent Vehicles*, 2024, 9(1): 1830–1846.
- [15] Zhao W, Shi K, Liu Z, Wu X, Zheng X, Wei L, Kato N. DRL Connects Lyapunov in Delay and Stability Optimization for Offloading Proactive Sensing Tasks of RSUs[J]. *IEEE Transactions on Mobile Computing*, 2024, 23(7): 7969–7982.
- [16] Qiu B, Wang Y, Xiao H, Zhang Z. Deep Reinforcement Learning-Based Adaptive Computation Offloading and Power Allocation in Vehicular Edge Computing Networks[J]. *IEEE Transactions on Intelligent Transportation Systems*, 2024, 25(10): 13339–13349.
- [17] Xie Y, Wu Q, Fan P. Digital twin vehicular edge computing network: Task offloading and resource allocation[C]. 2024 7th International Conference on Information Communication and Signal Processing (ICICSP). IEEE, 2024: 1137–1141.
- [18] Ale L, King S A, Zhang N, Sattar A R, Skandaraniyam J. D3PG: Dirichlet DDPG for Task Partitioning and Offloading With Constrained Hybrid Action Space in Mobile-Edge Computing[J]. *IEEE Internet of Things Journal*, 2022, 9(19): 19260–19272.
- [19] Zhang D-G, An H-Z, Zhang J, Zhang T, Dong W-M, Jiang X-R. Novel Privacy Awareness Task Offloading Approach Based on Privacy Entropy[J]. *IEEE Transactions on Network and Service Management*, 2024, 21(3): 3598–3608.
- [20] Wu G, Chen X, Gao Z, Zhang H, Yu S, Shen S. Privacy-preserving offloading scheme in multi-access mobile edge computing based on MADRL[J]. *Journal of Parallel and Distributed Computing*, 2024, 183: 104775.
- [21] Schaul T, Quan J, Antonoglou I, Silver D. Prioritized Experience Replay[A]. *arXiv*, 2016.