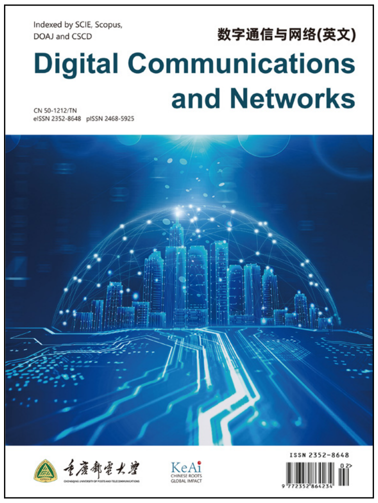
DRL-Based Federated Self-Supervised Learning for Task Offloading and Resource Allocation in ISAC-Enabled Vehicle Edge Computing

Xueying Gu, Qiong Wu, Pingyi Fan, Nan Cheng, Wen Chen et al.



PII:	S2352-8648(24)00175-5
DOI:	https://doi.org/10.1016/j.dcan.2024.12.009
Reference:	DCAN 839
To appear in:	*Digital Communications and Networks*
Received date:	20 August 2024
Revised date:	17 December 2024
Accepted date:	24 December 2024

Digital Communications and Networks(DCN)

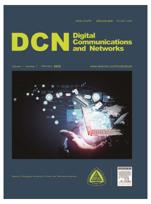
HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/dcan



DRL-Based Federated Self-Supervised Learning for Task Offloading and Resource Allocation in ISAC-Enabled Vehicle Edge Computing

Xueying Gu[a], Qiong Wu *[a], Pingyi Fan[b], Nan Cheng[c], Wen Chen[d], Khaled B. Letaief[e]

[a] *School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China*
[b]*Department of Electronic Engineering, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China*
[c]*State Key Lab. of ISN and School of Telecommunications Engineering, Xidian University, Xi'an 710071, China*
[d]*Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*
[e]*Department of Electrical and Computer Engineering, the Hong Kong University of Science and Technology (HKUST), Hong Kong 999077, China*

Abstract

Intelligent Transportation Systems (ITS) leverage Integrated Sensing and Communications (ISAC) to enhance data exchange between vehicles and infrastructure in the Internet of Vehicles (IoV). This integration inevitably increases computing demands, risking real-time system stability. Vehicle Edge Computing (VEC) addresses this by offloading tasks to Road Side Unit (RSU), ensuring timely services. Our previous work FLSimCo algorithm, which uses local resources for federated Self-Supervised Learning (SSL), though vehicles often can't complete all iterations task. Our improved algorithm offloads partial task to RSU and optimizes energy consumption by adjusting transmission power, CPU frequency, and task assignment ratios, balancing local and RSU-based training. Meanwhile, setting an offloading threshold further prevents inefficiencies. Simulation results show that the enhanced algorithm reduces energy consumption, improves offloading efficiency and the accuracy of federated SSL.

KEYWORDS:
Integrated sensing and communications (ISAC), Federated self-supervised learning, Resource allocation and offloading, Deep reinforcement learning (DRL), Vehicle edge computing (VEC)

1. Introduction

As Intelligent Transportation Systems (ITS) evolve, they increasingly rely on the integration of sensing and communication technologies, collectively known as Integrated Sensing and Communications (ISAC), to significantly improve the efficiency of data exchange between vehicles and infrastructure within the Internet of Vehicles (IoV) [1, 2]. However, this also brings about significant computing demands that often exceed local computing capability, potentially hindering real-time data processing [3, 4, 5]. To address these escalating computing demands, Vehicle Edge Computing (VEC) has emerged as a viable solution by deploying computing resources at the network edge, specifically at Road Side Units (RSUs) [6, 7, 8]. As integral components of the vehicular communication infrastructure, RSU is equipped with computing power and storage capacity, thereby delivering real-time computing services and ensuring timely task completion [9, 10, 11]. Meanwhile, offloading partial computing task to RSU can significantly reduce the computing burden on moving vehicles in IoV.

We previously proposed a Self-Supervised Learning (SSL) method based on Federated Learning (FL) called FLSimCo algorithm [12]. In each round of FL, each vehicle can calculate the expected number of local training iterations for SSL based on utilizing the entire computing resource of vehicle. However, for efficiency and practicality, vehicle often has multiple local tasks that need to process within a fixed time, including system maintenance, running other applications, and background processes [13, 14, 15]. This need for local task processing introduces competition for computing resources, complicating the execution of expected local training iterations. Due to CPU usage by other tasks, the expected local training iterations are often hard to fully execute in practice. In this case, offloading partial training iterations task to RSU can ensure the expected local training iterations task is completed within one round of FL [16, 17].

Compared to the extensive computing resources available in centralized cloud servers, the effective allocation of RSU resources becomes critical to meeting the demands of numerous task offloading requests from vehicles, particularly during peak periods [18, 19]. A simple solution is to deploy more RSUs, but this significantly increases costs of info-structure and may lead to substantial resource waste during off-peak periods. Therefore, this paper aims to maximize the utilization of a single RSU in a four-way roundabout, rationally allocate its computing resource, and meet more vehicle offloading requests [20, 21].

*Corresponding author.
[1]Email addresses: xueyinggu@stu.jiangnan.edu.cn(X. Gu), fpy@tsinghua.edu.cn(P. Fan), dr.nan.cheng@ieee.org(N. Cheng), wenchen@sjtu.edu.cn(W. Chen), eekhaled@ust.hk(K. B. Letaief)

In this paper, we further explore the potential of efficient offloading decisions based on the FLSimCo algorithm in IoV. We propose an improved algorithm that BS employs SAC algorithm to allocate transmission powers from vehicles to BS and RSU, CPU frequencies in vehicles, assignment ratios for computing resource in RSU. According to the allocated assignment ratio, the computing resource in RSU allowed for each vehicle can be calculated. Then according to the allowed computing resource, the local training iterations task in each round of FLSimCo is divided into local and RSU assisted training. However, at lower assignment ratio, it suggests the allocated computing resource in RSU is small. In this case, if vehicle offloads task to RSU, it does not ease the computing burden on vehicle and also increases communication costs on the requested tasks. Therefore, we also set an offloading threshold to optimize offloading efficiency. After offloading partial iterations task to RSU, the vehicle processes the remaining iterations task locally. If there are still unfinished iterations in round of FL, the remaining number of iterations will be stored in local buffer of vehicle, awaiting for the next training[23].

To summarize, the primary contributions of this paper are as follows:

- We propose a novel collaborative offloading method, based on the known FLSimCo algorithm, in the framework of federated SSL, which divides local iterations task into local and RSU assisted training in each round. By fully utilizing vehicle local computing resources and reasonably offloading partial task to RSU, significantly reduces the computing burden on moving vehicles in IoV.
- We use DRL algorithm for dynamic allocation of transmission power and assignment ratio to optimize system energy consumption. By adjusting transmission power and task assignment ratio, it can achieve minimizing energy consumption and improving the overall system efficiency.
- We introduce a task offloading threshold mechanism to enable rapid response under different network conditions and computing tasks, ensuring the efficiency of task offloading. This further enhances the utilization of computing resource on both the vehicle and RSU sides in IoV, increasing the efficiency and stability of the task offloading process.

The remaining sections of this paper are as follows. Section 2 provides a review of related work. Section 3 details the system model, including the computation model, channel model, and transmission model. In Section 4, we formulate an optimization problem for resource allocation. In Section 5, we introduce the DRL-based algorithm to solve the problem proposed in previous section. In Section 6, we present the simulation results, demonstrating the efficacy of our algorithm. Finally, in Section 7, we conclude the paper by summarizing the new findings.

[2]The source code has been released at: https://github.com/qiongwu86/Federated-SSL-task-offloading-and-resource-allocation

[3]This work was supported in part by the National Natural Science Foundation of China under Grant 61701197 and Grant 62071296; in part by the National Key Research and Development Program of China under Grant 2021YFA1000500(4); in part by the National Key Project under Grant 2020YFB1807700; in part by Shanghai Kewei under Grant 22JC1404000; in part by the Research Grants Council under the Areas of Excellence Scheme under Grant AoE/E-601/22-R; and in part by the 111 Project under Grant B23008.

2. Background and related work

In IoV, task offloading and computing resource allocation have been longstanding research hot spots [22, 23, 24]. With the increasing number of vehicles and the rising quantity and complexity of computing tasks, efficiently allocating and managing resource have become a focal point of attention. In this section, we will introduce the related work about task offloading and resource allocation.

2.1. DRL-based task offloading and resource allocation

With the rapid development of machine learning, an increasing number of work focus on applying machine learning to task offloading and resource allocation [25, 26]. Due to its superior performance in handling complex dynamic environments, DRL has gradually become a popular choice for optimizing task offloading and resource allocation.

Many work employed DRL methods for task allocation and resource allocation. In [27], Xu *et al.* proposed a mobile-compatible offloading and resource allocation scheme based on the DRL method (i.e., Deep Q-Network (DQN)) aiming to minimize system cost. In [28], Yu *et al.* proposed an optimization strategy for task offloading and resource allocation for static devices based on the DRL method (i.e., Deep Deterministic Policy Gradient (DDPG)), aiming to minimize system energy consumption and resource overhead. These methods are primarily designed for scenarios with limited mobility. In high-mobility scenarios, the aforementioned methods are not applicable because they do not consider the real-time changes of the target position and environment.

In [29], Chen *et al.* proposed an intelligent task offloading algorithm for Unmanned Aerial Vehicle (UAV) edge computing networks aimed at minimizing latency. The algorithm offloads all training tasks to edge servers and utilizes Deep Neural Networks (DNNs) for resource allocation. In [30], Wang *et al.* utilized an improved DRL method (i.e., DQN) to decide whether to execute tasks locally on vehicles or offload them to RSU in IoV. The goal is to achieve the lowest transmission and computation latency for both vehicles and RSU. In [31], Wang *et al.* proposed an offloading and resource allocation scheme in IoV based on the DRL method (i.e., Advantage Actor-Critic (A2C)), aiming to minimize latency and energy consumption.

Additionally, to improve offloading efficiency, some work has started using threshold to determine whether to offload. In [32], Chen *et al.* proposed that when the BS's offloading capacity is limited, setting an offloading threshold can significantly reduce signaling overhead. In [33], Al-Tuhafi and Al-Hemiary proposed an algorithm that dynamically adjusts the offloading threshold based on load increase or decrease to minimize response time. In [34], Qin *et al.* proposed a distributed threshold-based offloading algorithm. When the number of tasks on a device reaches the threshold, the tasks are offloaded to the cloud server for processing; otherwise, they are processed locally.

2.2. Task offloading in federated SSL

Some work combined FL and SSL [35, 36], but these methods mostly based on static environments, relying heavily on extensive negative samples during the SSL process, which leads to extended training duration. However, in real-time demanding environment such as IoV, constraints related to time and environmental variability become significant. Therefore, conducting

SSL in IoV necessitates addressing the challenge of reducing dependency on negative samples during the training process.

In [37], Wei *et al.* combined FL and SSL method (i.e., Momentum Contrast (MoCo)), and employed dictionary to reduce the demand for a large number of negative samples during training. However, this method requires sending each vehicle's small dictionary to the BS to form a large dictionary, which undoubtedly increases communication costs. In previous work, we proposed FLSimCo algorithm, a federated SSL algorithm without employing dictionary. However, in this algorithm, we also assume each vehicle can complete all local iterations task assigned by BS in each round. To enhance efficiency, vehicles need to handle multiple tasks per round, making it difficult to complete all iteration tasks in a fixed time. Offloading partial task to RSU can address this issue.

Meanwhile, some work has been done on DRL-based task offloading within the FL framework. In [38], Wu *et al.*, in the context of Industrial Internet of Things (IIoT), combined FL and DRL method to complete offloading task, aiming to improve offloading success rates. In [39], Zeng *et al.* proposed an online task offloading and resource allocation algorithm based on Federated DRL (i.e., DQN) in IoT, with the aim to regulate convergence speed, execution delay, overall computation rate and stability. In [40], Zhao *et al.* proposed an algorithm based on the FL and DRL method (i.e., Twin Delayed Deep Deterministic Policy Gradient (TD3)) in IIoT with Mobile Edge Computing (MEC). The aim is to minimize latency and energy consumption while maximizing security rates.

However, to the best of our knowledge, there is no existing work that utilizes DRL for threshold-based task offloading and resource allocation within the federated SSL framework.

3. System model

In this section, we will introduce the system scenario and models in this paper. As shown in Fig. 1, the scenario is an urban VEC traffic system containing four intersections. In this scenario, a BS is located at the corner, an RSU is deployed by the roadside, and N vehicles are moving on the roads. We assume that all four intersections are within the coverage area of both BS and RSU. When vehicle n, $n \in [1, N]$, arrives at an intersection, it chooses to turn left, turn right, or go straight through the intersection with probabilities of ψ_1, ψ_2, and ψ_3, respectively. The neural network configured on the BS is used for decision support. The RSU has fixed computing resource to parallel process tasks offloaded from vehicles [41, 42].

The entire process consists of two stages: offloading-based federated SSL and DRL-based resource allocation. These two stages are interdependent yet mutually influential. The stage of offload-based federated SSL consists E^{max} episodes, and each episode e, $e \in [1, E^{max}]$, is divided into R^{max} rounds. Similarly, the stage of DRL-based resource allocation also consists E^{max} episodes, and each episode e is divided into S^{max} slots [43, 44, 45]. Each round of FL corresponds to a slot, and the duration is T. However, when slot t reaches to S^{max}, the episode e enter to the next episode $e+1$, and slot t resets to zero. Meanwhile, the round r continues to accumulate until the entire algorithm is completed. Thus, the relationship between round r and slot t in episode e is described as

$$r = (e-1) \cdot S^{max} + t, \quad e \in [1, E^{max}] \text{ and } t \in [1, S^{max}] \tag{1}$$

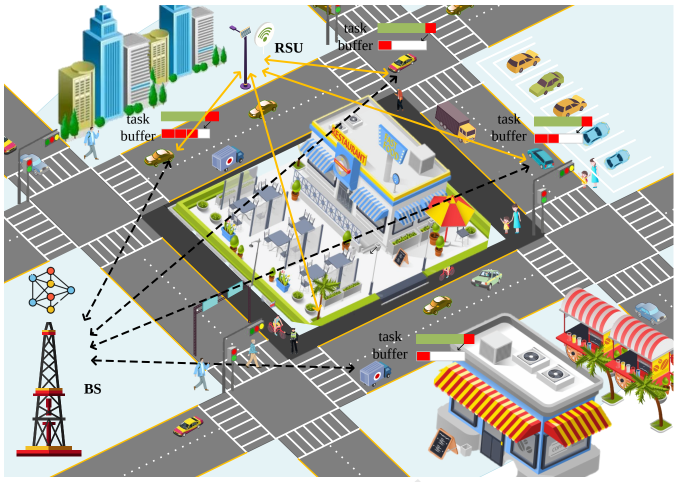


Fig. 1. System scenario

Following, we will introduce the computation model, channel model and transmission model in each round/slot.

3.1. Computation model

During the training process, the energy consumed for performing one training iteration on vehicle n is calculated as follows

$$E_{t,L}^{n,comp} = p_{t,L}^{n,comp} T_{t,L}^{n,comp} \tag{2}$$

where $p_{t,L}^{n,comp}$, $T_{t,L}^{n,comp}$, and $E_{t,L}^{n,comp}$ represent the computing power, delay, and energy consumption, respectively, for vehicle n to perform one local training iteration. The local CPU computation power $p_{t,L}^{n,comp}$ of vehicle n in slot t can be calculated using the Dynamic Voltage and Frequency Scaling (DVFS) method as follows

$$p_{t,L}^{n,comp} = \kappa \left(f_{t,L}^{n,comp}\right)^3 \tag{3}$$

where κ is the effective switching capacitance, which depends on the chip architecture. $f_{t,L}^{n,comp} \in \left[f^{min}, f^{max}\right]$, where f^{min} and f^{max} represent the minimum and maximum CPU frequency, respectively [46].

To calculate the computation delay of vehicle n, let c be the number of CPU cycles required to process a unit size of data. Z is the size of the local data, and it is assumed that each vehicle has the same data size. Then, cZ represents the number of CPU cycles needed for one iteration. Therefore, the computation delay $T_{t,L}^{n,comp}$ is calculated as follows

$$T_{t,L}^{n,comp} = \frac{cZ}{f_{t,L}^{n,comp}} \tag{4}$$

By substituting Eqs. (3) and (4) into Eq. (2), we obtain

$$E_{t,L}^{n,comp} = \kappa \left(f_{t,L}^{n,comp}\right)^2 cZ \tag{5}$$

Similarly, we can calculate the delay $T_{t,R}^{comp}$ and energy consumption $E_{t,R}^{comp}$ for each computation iteration at RSU side as

$$T_{t,R}^{comp} = \frac{cZ}{f_{t,R}^{comp}} \tag{6}$$

$$E_{t,R}^{comp} = \kappa \left(f_{t,R}^{comp}\right)^2 cZ \tag{7}$$

where $f_{t,R}^{comp}$ is the CPU frequency for computing of RSU in slot t.

3.2. Channel model

The vehicles involved in training use Orthogonal Frequency Division Multiple (OFDM) technology to communicate with the RSU [47, 48], so vehicle-to-vehicle interference is not considered during communication. Each vehicle uses the same channel model in the same slot, and the channel model is updated as the slot t advances to slot $t+1$ [49, 50]. Therefore, in slot t, the channel gain can be calculated as

$$h_{t,R}^{n,trans}\left[\text{dis}(R,n)\right] = \alpha_t^n\left[\text{dis}(R,n)\right] m_t^n \tag{8}$$

where m_t^n denotes small-scale fading, which follows an exponential distribution with a unit mean, i.e., $m_t^n \sim \text{Exp}(1)$. $\text{dis}(R,n)$ represents the distance between vehicle n and the RSU, and measured in meters. α_t^n represents the effect of large-scale fading on the signal, including path loss $\mathcal{P}$ and shadow fading $\mathcal{S}$. Thus, α_t^n can be calculated as

$$\alpha_t^n = 10^{\frac{\mathcal{P}}{10}} 10^{\frac{\mathcal{S}}{10}} \tag{9}$$

where shadow fading $\mathcal{S}$ follows a normal distribution with a mean of μ and a standard deviation of σ, represented as $\mathcal{S} \sim \mathcal{N}\left(\mu, \sigma^2\right)$. In high-density urban environments, the path loss $\mathcal{P}$ can be calculated as

$$\mathcal{P} = 128.1 + 37.6 \log\left[\frac{\text{dis}(R,n)}{1000}\right] \tag{10}$$

Therefore, the information transmission rate $R_{t,R}^n$ between the vehicle n and the RSU can be calculated as [51]

$$R_{t,R}^n = B^n \log_2\left[1 + \frac{p_{t,R}^n h_{t,R}^{n,trans} \text{dis}(R,n)}{N_0}\right] \tag{11}$$

where N_0 represents the noise power, and B^n denotes the bandwidth occupied by vehicle n when transmitting data to RSU.

3.3. Transmission model

If vehicle n offloads partial iterations task to the RSU, during the transmission process, the energy consumption $E_t^{n,trans}$ can be calculated as

$$E_t^{n,trans} = p_{t,R}^n T_{t,R}^{n,trans} \tag{12}$$

where $T_{t,R}^{n,trans}$ is the time required for vehicle n to transmit data to RSU. Assuming each vehicle transmits a local model of the same size D, and the data transmitted by vehicle n to RSU includes the untrained local model and local data. Therefore, the transmission delay $T_{t,R}^{n,trans}$ can be calculated as

$$T_{t,R}^{n,trans} = \frac{Z+D}{R_{t,R}^n} \tag{13}$$

Thus, the transmission energy consumption $E_t^{n,trans}$ from vehicle n to RSU can be calculated as

$$E_t^{n,trans} = \frac{p_{t,R}^n (Z+D)}{R_{t,R}^n} \tag{14}$$

4. Optimization problem

When vehicles are performing local iterations task, if they are unable to complete all iterations locally due to local computing resource limitations, partial iterations task needs to be offloaded to RSU [52]. This offloading process requires BS to use a DRL algorithm to allocate assignment ratios, which determines the computing resource of the RSU allowed for each vehicle. The total iterations are then allocated between local training and training on the RSU. This optimization aims to enhance resource utilization and task completion efficiency. Next, we will delve into the specific methods of task allocation.

4.1. Task allocation

Within slot t, vehicle n is able to determine the total local training iterations N_t^n, i.e.,

$$N_t^n = \frac{T - t^{max}}{T_{t,L}^{n,comp}} \tag{15}$$

where t^{max} represents the maximum time used for data transmission. Thus, $T - t^{max}$ indicates the total time available for computing within a slot. In practical, the CPU typically handles multiple tasks simultaneously to improve CPU utilization. These tasks consume some CPU computing resources, resulting in the actual time available for local iterations being less than T. We denote the true training time as T'. Therefore, the actual total local training iterations N_t^{n*} can be calculated as

$$N_t^{n*} = \frac{T' - t^{max}}{T_{t,L}^{n,comp}} \tag{16}$$

Substitute Eq. (4) into Eq. (15) and Eq. (16), the final results can be obtained, respectively.

$$N_t^n = \frac{(T - t^{max}) f_{t,L}^{n,comp}}{cZ} \tag{17}$$

$$N_t^{n*} = \frac{(T' - t^{max}) f_{t,L}^{n,comp}}{cZ} \tag{18}$$

In this paper, we study a binary partial offloading problem, where a binary variable g_t^n is used to indicate whether vehicle n needs to offload partial task to the RSU. g_t^n is related to the assignment ratio q_t^n and the assignment ratio threshold q_0. When q_t^n is less than the assignment ratio threshold q_0, it indicates that vehicle n will not offload partial task to RSU and reset $q_t^n = 0$; otherwise, it will offload it. Therefore, g_t^n can be expressed as

$$g_t^n = \begin{cases} 1, & \text{if } q_t^n \geq q_0 \\ 0, & \text{if } q_t^n < q_0 \end{cases} \tag{19}$$

We denote the total task that vehicle n needs to handle in slot t as $N_t^{n,total}$, which includes the sum of the total training iterations N_t^n in slot t and the remaining unprocessed training iterations B_{t-1}^n from the previous slot. When vehicle n offloads partial task to RSU, the total task $N_t^{n,total}$ will be divided into local training iterations $N_{t,L}^n$ and offloaded training iterations $N_{t,R}^{n,off}$ according to the allocated assignment ratio q_t^n. Specifically, RSU first determines the total training iterations N_R in RSU side as follows

$$N_R = \frac{T - t^{max} - T_{t,R}^{n,trans}}{T_{t,R}^{comp}} \tag{20}$$

Thus, the expected offloaded training iterations $N_{t,R}^{n,off}$ for RSU can be calculated as

$$N_{t,R}^{n,off} = g_t^n q_t^n N_R \tag{21}$$

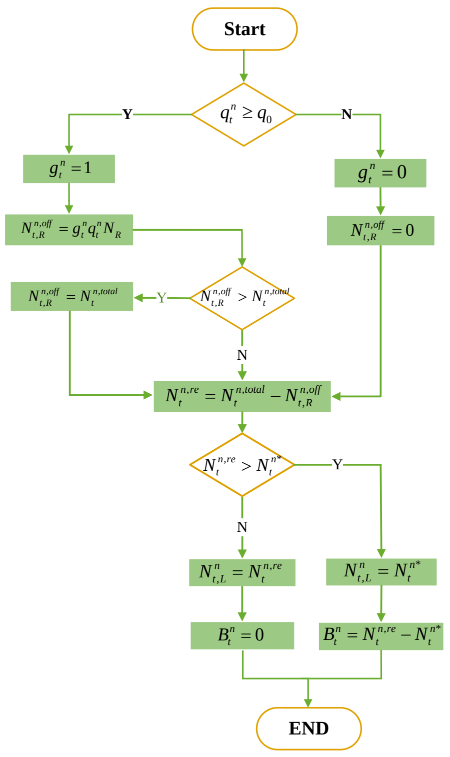

Start
$q_t^n \geq q_0$
Y
N
$g_t^n = 1$
$g_t^n = 0$
$N_{t,R}^{n,off} = g_t^n q_t^n N_R$
$N_{t,R}^{n,off} = 0$
$N_{t,R}^{n,off} = N_t^{n,total}$
Y
$N_{t,R}^{n,off} > N_t^{n,total}$
N
$N_t^{n,re} = N_t^{n,total} - N_{t,R}^{n,off}$
$N_t^{n,re} > N_t^{n*}$
Y
N
$N_{t,L}^n = N_t^{n,re}$
$N_{t,L}^n = N_t^{n*}$
$B_t^n = 0$
$B_t^n = N_t^{n,re} - N_t^{n*}$
END

Fig. 2. Task allocation flowchart

If $N_{t,R}^{n,off}$ exceeds $N_t^{n,total}$, it indicates that the expected offloading task is too large, surpassing the total task that need to be processed locally. In this case, the actual offloaded training iterations are recorded as $N_t^{n,total}$. Thus, $N_{t,R}^{n,off}$ can be updated as

$$N_{t,R}^{n,off} \leftarrow \min\left(N_{t,R}^{n,off}, N_t^{n,total}\right) \tag{22}$$

So far, we have obtained the actual offloading iterations $N_{t,R}^{n,off}$. Following, we will calculate the number of local iterations $N_{t,L}^n$.

The remaining portion of $N_t^{n,total}$ denoted as $N_t^{n,re}$, which represents the number of iterations that need to be processed locally on the vehicle n, and can be calculated as

$$N_t^{n,re} = N_t^{n,total} - N_{t,R}^{n,off} \tag{23}$$

For the local computing capability of the vehicle n, its actual total number of training iterations is N_t^{n*}. When $N_t^{n,re}$ does not exceed N_t^{n*}, vehicle n can handle all remaining task locally, i.e., $N_{t,L}^n = N_t^{n,re}$. When $N_t^{n,re}$ exceeds N_t^{n*}, it means vehicle n cannot handle all remaining task locally, and the maximum iterations can be deal with locally are N_t^{n*}. Meanwhile, the unprocessed portion is placed in the local buffer B_t^n. We abbreviate $N_{t,L}^n$ as

$$N_{t,L}^n = \min\left(N_t^{n,re}, N_t^{n*}\right) \tag{24}$$

and B_t^n is expressed as

$$B_t^n = \begin{cases} 0, & \text{if } N_t^{n,re} \leq N_t^{n*} \\ N_t^{n,re} - N_t^{n*}, & \text{if } N_t^{n,re} > N_t^{n*} \end{cases} \tag{25}$$

The specific process of task allocation is as shown in Fig. 2, and the involved symbols for vehicle n in slot t are listed in Table 1.

Table 1
Symbols and meanings.

Symbol	Meaning
$p_{t,B}^n$	Transmission power from vehicle to BS
$p_{t,R}^n$	Transmission power from vehicle to RSU
$p_{t,L}^{n,comp}$	Computing power for vehicle
$f_{t,L}^{n,comp}$	CPU frequency for vehicle
$f_{t,R}^{comp}$	CPU frequency for RSU
N_t^n	Expected total number of training iterations
N_t^{n*}	Actual total number of training iterations
$N_{t,L}^n$	Number of local iterations
$N_{t,R}^{n,off}$	Offloaded training iterations
N_R	Total training iterations in RSU
$N_t^{n,total}$	Total task
$N_t^{n,re}$	Remaining portion of total task
$T_{t,L}^{n,comp}$	Delay for each computation iteration at vehicle
$T_{t,R}^{comp}$	Delay for each computation iteration at RSU
$T_{t,R}^{n,trans}$	Transmission delay from vehicle to RSU
$h_{t,R}^{n,trans}$	Channel gain
q_t^n	Assignment ratio
m_t^n	Small-scale fading
α_t^n	Large-scale fading
$E_{t,L}^{n,comp}$	Energy consumption for each computation iteration at vehicle
$E_{t,R}^{comp}$	Energy consumption for each computation iteration at RSU
$E_t^{n,trans}$	Energy consumption for transmission from vehicle to RSU
$R_{t,R}^n$	Information transmission rate between the vehicle and RSU
g_t^n	If offloading
B_{t-1}^n	Remaining unprocessed training iterations from the previous slot

4.2. *Energy consumption*

Increasing the number of local iterations typically enhances classification accuracy, but it also leads to increased energy consumption per round of FL. To find a balance between classification accuracy and energy consumption, we shall formulate an optimization model to find its solution.

Due to the self-interested nature of individual vehicles, their decisions often aim to maximize their own benefits. However, this self-interested behavior can lead to imbalances in system resource utilization efficiency and energy consumption. In contrast, a key advantage of centralized allocation by BS is its ability to balance overall system resource utilization efficiency and energy consumption. In this section, we propose an optimization algorithm aimed at minimizing the energy consumption of the entire system. The energy consumption is divided into three parts: RSU computing energy consumption $E_{t,R}^n$, transmission energy consumption $E_t^{n,trans}$ from vehicle n to RSU, and local computing energy consumption $E_{t,L}^n$. $E_t^{n,trans}$ can be calculated according to Eq. (14). $E_{t,R}^n$ and $E_{t,L}^n$ are represented as, respectively

$$E_{t,R}^n = E_{t,R}^{comp} N_{t,R}^{n,off} \tag{26}$$

$$E_{t,L}^n = E_{t,L}^{n,comp} N_{t,L}^n \tag{27}$$

Therefore, the total energy consumption $E_t^{n,total}$ for each vehicle n in slot t can be expressed as

$$E_t^{n,total} = g_t^n E_{t,R}^n + E_{t,L}^n + g_t^n E_t^{n,trans} \tag{28}$$

4.3. *Problem formulation*

To achieve efficient task offloading and resource allocation, it is essential for BS to consider the long-term impact of their actions during the decision-making process. Therefore, we aim to minimize the sum of the energy consumption of all vehicles in each slot. Let $\mathbb{P} = \left\{p_{t,R}^1, p_{t,R}^2, \cdots p_{t,R}^n, \cdots p_{t,R}^N\right\}$, $\mathbb{F} = \left\{f_{t,L}^{1,comp}, f_{t,L}^{2,comp}, \cdots f_{t,L}^{n,comp}, \cdots f_{t,L}^{N,comp}\right\}$ and $\mathbb{Q} =$

$\left\{q_t^1, q_t^2, \cdots q_t^n, \cdots q_t^N\right\}$. Then, we can formulate the optimization problem in slot t as follows

$$\min_{\mathbb{P},\mathbb{F},\mathbb{Q}} : \sum_{n=1}^{N} E_t^{n,total} \tag{29}$$

s.t.

$$g_t^n \in \{0,1\}, \forall n \in [1,N] \tag{29a}$$

$$p^{min} \le p_{t,R}^n \le p^{max}, \forall n \in [1,N] \tag{29b}$$

$$f^{min} \le f_{t,L}^{n,comp} \le f^{max}, \forall n \in [1,N] \tag{29c}$$

$$0 \le q_t^n \le 1, \forall n \in [1,N] \tag{29d}$$

$$\sum_{n=1}^{N} q_t^n \le 1 \tag{29e}$$

Eq. (29a) signifies that the offloading task is formulated as a binary decision problem, i.e., whether to offload or not. p^{min} and p^{max} is the the minimum and maximum transmission power. Eq. (29b) restricts the transmission power to the range between p^{min} and p^{max}, while Eq. (29c) limits the CPU frequency of vehicle n to the range between f^{min} and f^{max}. Eqs. (29d) and (29e) constrain the assignment ratio, ensuring that the number of offloaded tasks does not exceed the computing capacity of RSU.

4.4. Evaluation metrics

Evaluation metrics quantify the system's performance, ensuring effective and objective assessment and optimization of task offloading and resource allocation scheme. These metrics enable fair comparison across different algorithms, thereby enhancing the overall system performance and efficiency. In this section, we will introduce the evaluation metrics used in our paper.

4.4.1. Overload ratio

After the BS allocates the assignment ratio q_t^n for vehicle n in slot t, the iterations number of offloading task $N_t^{n,off}$ assigned to vehicle n might exceed the total number of tasks $N_t^{n,total}$ that the vehicle n needs to process in slot t [28]. We denote the excess part as the overload O_t^n, which can be calculated as

$$O_t^n = \max\left(0, N_{t,R}^{n,off} - N_t^{n,total}\right) \tag{31}$$

We use the overload ratio $\mathcal{R}_0$ to describe the average severity of the overload behavior across all vehicles over S^{max} slots, that is,

$$\mathcal{R}_0 = \frac{1}{NS^{max}}\left(\sum_{n=1}^{N}\sum_{t=1}^{S^{max}} \frac{O_t^n}{N_t^{n,total}}\right) \tag{32}$$

4.4.2. Offloading efficiency

As described above, when the assignment ratio q_t^n is below a specific threshold q_0, vehicle n will not perform offloading operations. Therefore, we define offloading efficiency η_{q_0} as a metric to measure the system's capability to offload tasks, i.e.,

$$\eta_{q_0} = \frac{1}{NS^{max}}\sum_{n=1}^{N}\sum_{t=1}^{S^{max}}\left(\frac{N_t^{n,off}}{N_t^{n,total}} \times 100\%\right) \tag{33}$$

5. DRL-based solution

The problem of task offloading and resource allocation can be addressed with a DRL algorithm. To this end, we will utilize DRL to find the optimal task offloading and resource allocation scheme. The SAC algorithm is chosen due to its robustness and efficiency in handling continuous action spaces and its ability to balance exploration and exploitation effectively. SAC is an off-policy DRL algorithm designed to enhance the stability of learning and the exploration of the policy. It encourages exploration through the maximum entropy framework, preventing premature convergence to suboptimal solutions. Additionally, SAC uses DQN to reduce overestimation bias in value function approximation, further improving the stability. Compared to other DRL algorithms, SAC introduces an entropy term into the objective function to encourage BS to continuously explore new policies. The presence of this entropy term effectively prevents BS from prematurely converging to a local optimal solution during policy iteration, thereby preventing training failure.

We employ SAC algorithm with the objective of minimizing the overall system's energy consumption, aiming to find an optimal resource allocation scheme. Next, we will introduce this algorithm.

For the decision-making process in each slot t, it includes state, action, reward, and solution. We define the state, action, reward, and solution in slot t as follows

5.1. State

In this case, BS is able to analysis state and experiences during the process of decision-making. The system state consists of two elements: the set of gain information $\mathbb{G} = \left\{h_{t,R}^{1,trans}, h_{t,R}^{2,trans}, \cdots h_{t,R}^{n,trans}, \cdots h_{t,R}^{N,trans}\right\}$, the set of vehicle velocities $\mathbb{V} = \left\{v_t^1, v_t^2, \cdots v_t^n, \cdots v_t^N\right\}$ and the total task $\mathbb{N} = \left\{N_t^{1,total}, N_t^{2,total}, \cdots N_t^{n,total}, \cdots N_t^{N,total}\right\}$. Here, the gain information h_t^n is determined based on the current position of vehicle n, and the velocity v_t^n, $v_t^n \in [v^{min}, v^{max}]$, remains constant within each slot and changes at the beginning of next slot. v^{min} and v^{max} respectively denote the minimum and maximum values of vehicle velocity. Therefore, the state can be expressed as

$$s_t = (\mathbb{G}, \mathbb{V}, \mathbb{N}) \tag{34}$$

5.2. Action

After obtaining the state, the BS makes decisions based on SAC algorithm for slot t. We set the action as three discrete variables: the set of powers $\mathbb{P} = \left\{p_{t,R}^1, p_{t,R}^2, \cdots p_{t,R}^n, \cdots p_{t,R}^N\right\}$ when the vehicles send data to RSU, the set of CPU computation frequencies $\mathbb{F} = \left\{f_{t,L}^{1,comp}, f_{t,L}^{2,comp}, \cdots f_{t,L}^{n,comp}, \cdots f_{t,L}^{N,comp}\right\}$, and the set of assignment ratios $\mathbb{Q} = \left\{q_t^1, q_t^2, \cdots q_t^n, \cdots q_t^N\right\}$. Thus, the action in slot t can be expressed as

$$a_t = (\mathbb{P}, \mathbb{F}, \mathbb{Q}) \tag{35}$$

5.3. Reward

Our objective is to minimize energy consumption, overload amount, and the remaining local iterations in the buffer within each slot. Thus, in slot t, the reward consists of three negative components: the total energy consumption $E_t^{n,total}$, the overload amount O_t^n, and the remaining local iterations B_t^n at the end of slot t. The reward function can be calculated as

$$r_t = -\sum_{n=1}^{N}\left(\vartheta_1 E_t^{n,total} + \vartheta_2 O_t^n + \vartheta_3 B_t^n\right) \tag{36}$$

where ϑ_1, ϑ_2, and ϑ_3 represent weight of the penalty term.

5.4. SAC-based task offloading and resource allocation solution in federated SSL

The SAC algorithm bases on the maximum entropy principle, and it involves five networks: one actor network ς, two critic networks ξ_1 and ξ_2, and two target networks ϖ_1 and ϖ_2. At the start of the scheme, the BS holds the parameters of the global model, while each vehicle retains a local model that has the same structure as the global model. Following, we will introduce the complete SAC-based federated SSL solution for task offloading and resource allocation.

Step 1, initialization: When episode $e = 1$, if in the first round/slot, the BS randomly initializes the global model, as well as the actor and critic networks within the SAC algorithm, and assigns the parameters of the critic networks to two target networks.

Step 2, task offloading and training: In episode e, before slot t starts, determine the relationship between round r and slot t according to Eq. (1) firstly. BS stores the parameters of a global model $\theta_{r-1}^{\text{global}}$ aggregated in previous round $r-1$, along with the five networks of SAC algorithm. If in episode e, slot 1, the parameter of five networks come from episode $e-1$, slot S^{max}. Vehicles also store $\mathcal{Z}$ image data, denoted as local data, and a local model which has the same structure as global model. Then, BS employs SAC algorithm to allocate $p_{t,R}^n$, $f_{t,L}^{n,comp}$, and q_t^n for vehicle n. Next, vehicle n downloads the global model $\theta_{r-1}^{\text{global}}$ from BS and assigns the parameters of global model to local model $\theta_r^{\text{local},n}$, along with the transmission power $p_{t,R}^n$, $f_{t,L}^{n,comp}$, and q_t^n allocated by BS for vehicle n. Based on the downloaded CPU frequency $f_{t,L}^{n,comp}$, the total training time T, and true training time T', N_t^n and N_t^{n*} can be calculated by Eq. (17) and Eq. (18). Then, conducting task allocating to obtain $N_{t,L}^n$ and $N_{t,R}^{n,off}$ according to Section 4.1. If the downloaded assignment ratio q_t^n is greater than the offloading threshold q_0, the vehicle n immediately sends an offloading request to RSU with the downloaded transmission power $p_{t,R}^n$. This request includes local data $\left\{x_r^{1,n},\ x_r^{2,n},\ \ldots x_r^{i,n}, \ldots x_r^{\mathcal{Z},n}\right\}$, local model $\theta_r^{\text{local},n}$, and assignment ratio q_t^n.

Meanwhile, vehicle n also performs local training with local data and local model for $N_{t,L}^n$ iterations. The training process is a federated SSL. For each iteration, first, input each local data $x_r^{i,n}$, $x_r^{i,n} \in \left\{x_r^{1,n},\ x_r^{2,n},\ \ldots x_r^{i,n}, \ldots x_r^{\mathcal{Z},n}\right\}$, into two different data augmentation methods, $\pi_1(\cdot)$ and $\pi_2(\cdot)$. The function $\pi_1(\cdot)$ applies a horizontal flip to the image with a 50% likelihood, and converts the image to grayscale with a 20% likelihood. Meanwhile, $\pi_2(\cdot)$ adjusts the brightness of image, saturation, and hue with an 80% likelihood, and then converts the image to grayscale with a 40% likelihood. These two different augmentations are adopted in the FLSimCo, so the features of image can be efficiently utilized.

The remaining images $x_r^{j,n}$, $j \in [1, \mathcal{Z}]$ and $j \neq i$ are denoted as negative samples. Feeding augmented images $x_r^{i,n}$ and $x_r^{j,n}$ into a function χ, which is composed of an encoder, two dense neutral networks and a ReLU function. The output of the χ includes anchor sample $q_r^{i,n}$, positive sample $k_r^{i,n}$, and encoded negative samples $k_r^{j,n}$, respectively, i.e.,

$$q_r^{i,n} = \chi\left\{\pi_1\left(x_r^{i,n}\right)\right\},\ i \in [1, \mathcal{Z}] \tag{37}$$

$$k_r^{i,n} = \chi\left\{\pi_2\left(x_r^{i,n}\right)\right\},\ i \in [1, \mathcal{Z}] \tag{38}$$

$$k_r^{j,n} = \chi\left\{x_r^{j,n}\right\},\ j \in [1, \mathcal{Z}],\ \text{and } j \neq i \tag{39}$$

The loss can be calculated by Information Noise Contrastive Estimation (InforNCE) as [53]

$$\mathcal{L}_{q_r^{i,n}|\tau=\tau_1} = \frac{\exp\left(\frac{q_r^{i,n}\cdot k_r^{i,n}}{\tau}\right)}{\exp\left(\frac{q_r^{i,n}\cdot k_r^{i,n}}{\tau}\right) + \sum_{j=1}^{\mathcal{K}} \exp\left(\frac{q_r^{i,n}\cdot k_r^{j,n}}{\tau}\right)} \tag{40}$$

where $\mathcal{K}$ is the number of negative samples, and τ is temperature hyper-parameter [54]. Thus, the dual temperature loss used by FLSimCo can be calculated as

$$\mathcal{L}_{q_r^{i,n}}^{\text{DT}} = \text{sg}\left[\frac{1-\mathcal{L}_{q_r^{i,n}|\tau=\tau_2}}{1-\mathcal{L}_{q_r^{i,n}|\tau=\tau_1}}\right] \times \log \mathcal{L}_{q_r^{i,n}|\tau=\tau_1} \tag{41}$$

where sg[·] indicates the stop gradient. τ_1 and τ_2 are different values of temperature hyper-parameter τ. We use the Stochastic Gradient Descent (SGD) method to update the local model [55], i.e.,

$$\theta_{r,L}^{\text{local},n} = \theta_r^{\text{local},n} - \eta^r\left[\nabla_{q_r^{i,n}}\left(\frac{1}{\mathcal{Z}}\sum_{i=1}^{\mathcal{Z}} \mathcal{L}_{q_r^{i,n}}^{\text{DT}}\right)\right] \tag{42}$$

where η^r is the learning rate and $\theta_{r,L}^{\text{local},n}$ is the model trained locally by vehicle n in round r. Noting that, during the process of local training, vehicle n also captures images $\left\{x_{r+1}^{1,n},\ x_{r+1}^{2,n},\ \ldots x_{r+1}^{i,n}, \ldots x_{r+1}^{\mathcal{Z},n}\right\}$, as local data, for next round.

At the same time, upon receiving the vehicle's offloading request, RSU performs training for $N_{t,R}^{n,off}$ iterations using the local data and local model upload by vehicle n to generate a new model $\theta_{r,R}^{\text{local},n}$, similar with the training process as vehicle locally training. After the training is completed, the RSU sends the trained model $\left\{\theta_{r,R}^{\text{local},1},\ \theta_{r,R}^{\text{local},2},\ \ldots \theta_{r,R}^{\text{local},n}, \ldots \theta_{r,R}^{\text{local},N}\right\}$ back to each vehicle. Based on $N_{t,L}^n$ and $N_{t,R}^{n,off}$, the total energy consumption $E_t^{n,total}$ can be calculated according to Eq. (28).

Step 3, uploading: After receiving all trained model sent by RSU, vehicles upload trained models $\left\{\theta_{r,L}^{local,1}, \theta_{r,L}^{local,2}, \cdots \theta_{r,L}^{local,n}, \cdots \theta_{r,L}^{local,N}\right\}$, $\left\{\theta_{r,R}^{local,1}, \theta_{r,R}^{local,2}, \cdots \theta_{r,R}^{local,n}, \cdots \theta_{r,R}^{local,N}\right\}$, velocities $\left\{v_{t+1}^1, v_{t+1}^2, \cdots v_{t+1}^n, \cdots v_{t+1}^N\right\}$ for slot $t+1$, total energy consumption $\left\{E_t^{1,total}, E_t^{2,total}, \cdots E_t^{n,total}, \cdots E_t^{N,total}\right\}$, and unprocessed training iterations $\left\{B_t^1, B_t^2, \cdots B_t^n, \cdots B_t^N\right\}$ to BS, respectively.

Step 4, aggregating and updating: Upon receiving the models from all vehicles, the BS aggregates received models followed as

$$\theta_r^{\text{global}} = \frac{\sum_{n=1}^{N}\left(\theta_{r,L}^{\text{local},n} + \theta_{r,R}^{\text{local},n}\right)}{2N} \tag{43}$$

where θ_r^{global} is the parameters of new global model in round r. Simultaneously, the reward is calculated by Eq. (36), and the network is updated accordingly. Next, we will introduce the process of updating the SAC networks.

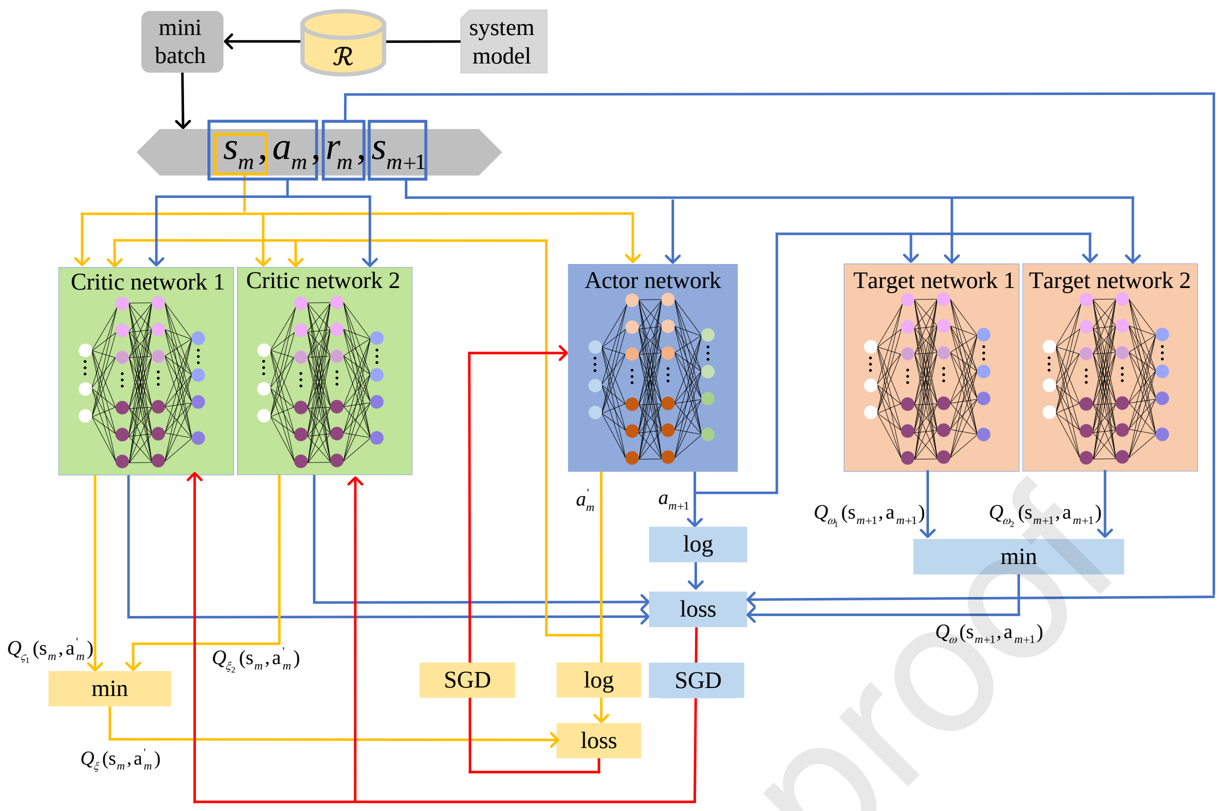


Fig. 3. The process of updating actor and critic networks of SAC algorithm

The core idea of the SAC algorithm is to balance the exploration-exploitation trade-off by increasing the stochasticity of policy based on entropy. The objective function $J(\pi_\varsigma)$ of SAC can be expressed as the weighted sum of expected reward and entropy, i.e.,

$$J(\pi_\varsigma) = E_{(s_t,a_t)\sim\pi_\varsigma}\left[\gamma^{t-1} r_t \ + \beta H(\pi_\varsigma(\cdot|s_t))\right] \tag{44}$$

where $H(\pi_\varsigma(\cdot|s_t))$ denotes the entropy of actor network policy π_ς in state s_t. $\gamma \in [0, 1]$ is the discount factor, and β is the temperature parameter that adjusts the randomness of the optimal policy, balancing the importance of the expected reward and the entropy.

Our goal is to maximize $J(\pi_\varsigma)$, and upon achieving maximum $J(\pi_\varsigma)$, the corresponding policy π_ς is denoted as π_ς^*. Using π_ς^*, we can adjust β to obtain the optimal solution under state s_t, which is represented as β^*. Thus, β^* can be expressed as

$$\beta^* = \arg\max_{\beta} J(\pi_\varsigma^*) \tag{45}$$

BS obtains the initial state s_t from the system, then selects the action a_t based on the policy of actor network π_ς. After executing action a_t, the state s_t transitions to a new state s_{t+1} and receives the corresponding reward r_t. These interaction data are stored in the replay buffer R in the form of tuples (s_t, a_t, r_t, s_{t+1}). The replay buffer R allows for random sampling of these tuples, breaking the correlation between samples during training. When the number of tuples stored in the replay buffer R reaches to batch size $\mathcal{B}$, begin to update five networks of SAC algorithm.

Randomly sample M tuples from the replay buffer R, referred to as a mini batch. For each tuple (s_m, a_m, r_m, s_{m+1}) in the mini batch, where $m \in [1, M]$, feed s_m into the actor network ς. Based on the policy π_ς of the actor network ς, a new action a'_m is obtained. The loss function of β can be calculated as

$$\nabla_\beta J_\beta = -\nabla_\beta \ E_{a'_m\sim\pi_\varsigma}\left[\beta \log \pi_\varsigma(a'_m|s_m) + \beta\widetilde{H}\right] \tag{46}$$

where, $\widetilde{H}$ represents the dimensions of action a_t.

Next, as shown by the yellow line in Fig. 3, we start to update the actor network ς. Firstly, feeding (s_m, a'_m) into the two critic networks ξ_1 and ξ_2, respectively, yielding the corresponding action value functions $Q_{\xi_1}(s_m, a'_m)$ and $Q_{\xi_2}(s_m, a'_m)$. The smaller of these two values is defined as $Q_\xi(s_m, a'_m)$. Thus, by the SGD algorithm, the actor network ς is updated as follows

$$\begin{aligned}\nabla_\varsigma \mathcal{J}_\varsigma = \nabla_\varsigma &\left[-\frac{1}{M}\sum_{m=1}^{M}\left(\beta \log \pi_\varsigma\left(a'_m\middle|s_m\right)\right)^2\right] \\ &+ \nabla_\varsigma\left[-\frac{1}{M}\sum_{m=1}^{M}\left(f_\varsigma(o;s_m) \times \nabla_{a'_m}\left(Q_\xi(s_m, a'_m)\right.\right.\right. \\ &\left.\left.\left.-\beta \log \pi_\varsigma\left(a'_m\middle|s_m\right)\right)\right)^2\right]\end{aligned} \tag{47}$$

where o is noise sampled from a multivariate normal distribution, and $f_\varsigma(o; s_m)$ is a function utilized to re-parameterize the action a'_m [56].

Following, as the blue line shown in Fig. 3, to update the critic networks, first input (s_m, a_m) into both critic networks ξ_1 and ξ_2, and obtain the action value pairs $Q_{\xi_1}(s_m, a_m)$ and $Q_{\xi_2}(s_m, a_m)$, respectively. Meanwhile, input s_{m+1} into the actor network ς, and obtain a_{m+1} under the policy π_ς. To maintain randomness in the policy π_ς during training and explore a wider range of states and actions, the SAC algorithm introduces an entropy regularization term to the policy function, namely $\log\pi_\varsigma^{m+1}(a_{m+1}|s_{m+1})$. Then, input (s_{m+1}, a_{m+1}) into the two target networks ϖ_1 and

ϖ_2, and obtain the target action value pairs $Q_{\varpi_1}(s_{m+1}, a_{m+1})$ and $Q_{\varpi_2}(s_{m+1}, a_{m+1})$. Take the smaller of the two values, denoted as $Q_{\varpi}(s_{m+1}, a_{m+1})$. Finally, updating the critic networks by the SGD algorithm, i.e.,

$$\begin{aligned}\nabla_{\xi_1} J_{\xi_1} &= \nabla_{\xi_1} Q_{\xi_1}(s_m, a_m) \times \Big[Q_{\xi_1}(s_m, a_m) - r_m \\ &+\gamma[-\beta \log \pi_{\varsigma}^{m+1}(a_{m+1}|s_{m+1}) + Q_{\varpi}(s_{m+1}, a_{m+1})]\Big]\end{aligned} \tag{48}$$

$$\begin{aligned}\nabla_{\xi_2} J_{\xi_2} &= \nabla_{\xi_2} Q_{\xi_2}(s_m, a_m) \times \Big[Q_{\xi_2}(s_m, a_m) - r_m \\ &+\gamma[-\beta \log \pi_{\varsigma}^{m+1}(a_{m+1}|s_{m+1}) + Q_{\varpi}(s_{m+1}, a_{m+1})]\Big]\end{aligned} \tag{49}$$

We adopt the Adam optimizer to update the actor and critic networks every I_R slots based on $\nabla_\beta J_\beta$, $\nabla_\varsigma \mathcal{J}_\varsigma$, $\nabla_{\xi_1} J_{\xi_1}$ and $\nabla_{\xi_2} J_{\xi_2}$ in Eqs. (46) - (49). And now, the process of updating actor and critic networks in slot t are finished and enter to the next slot. Meanwhile, in the process, every I_t slots, we also need to update the parameters of the two target networks ϖ_1 and ϖ_2 as follows

$$\varpi_1 := \omega\xi_1 + (1-\omega)\varpi_1 \tag{50}$$

$$\varpi_2 := \omega\xi_2 + (1-\omega)\varpi_2 \tag{51}$$

where $\omega \ll 1$.

Repeat ***Step 2*** to ***Step 4*** until episode e reaches to E^{max} and slot t reaches to S^{max}, at which point the entire algorithm terminates. Then output a final global model and policy of actor network, denoted as ς^*.

6. Simulation results

6.1. Base settings

This experiment is entirely based on the scenario described in Section 3, and utilizes Python 3.10 for simulation. The actor network and both critic networks are implemented as four-layer fully connected DNNs, each featuring two hidden layer sand containing 512 neurons. We set up a scenario with four intersections. In urban environments, the variety of object categories is relatively limited, and images of objects within the same category are frequently captured. Therefore, we chose the CIFAR-10 dataset, which contains 10 common categories with 5,000 distinct images per category.

The following training simulation results represent the average of three experiments, and the environmental parameters and SAC algorithm hyper-parameters used in this paper are shown in Table 2.

Table 2
Hyper-parameter.

Parameter	**Value**	**Parameter**	**Value**
$\vartheta_1/\vartheta_2/\vartheta_3$	10/0.001/0.01	$\psi_1/\psi_2/\psi_3$	0.3/0.3/0.4
E^{max}	3000	S^{max}	100
T	1s	T'	0.005T~ T
Z	1500KB	$\mathcal{Z}$	512
f^{min}	5×10^7Hz	f^{max}	4×10^8Hz
p^{min}	5W	p^{max}	200W
v^{min}	10m/s	v^{max}	15m/s
τ_1	0.1	τ_2	1
μ	0	σ	8
m	0.023	κ	10^{-27}
$\mathcal{B}$	256	M	64
N_0	-114dB	$\mathcal{N}$	512
D	11.2M	t^{max}	0.02
I_R	2	I_t	80
c	1600 cyc/s	R	10^6
Initial learning rate	0.01	momentum of SGD	0.9
$f_{t,R}^{comp}$	6×10^9Hz	γ	0.99
q_0	0.005	total bandwidth	2×10^6Hz

6.2. Comparative algorithms

To conduct comparative experiments, we selected three additional DRL algorithms suitable for continuous action spaces: DDPG, TD3, and Proximal Policy Optimization (PPO).

- DDPG: DDPG is a deterministic policy gradient algorithm based on deep learning. It combines the advantages of policy gradient methods and Q-learning, allowing for direct learning of the policy function without the need to explicitly learn the value function.

- TD3: TD3 is an improvement and extension of DDPG. This algorithm reduces the variance in Q-value estimation by introducing twin Q-networks. Additionally, it improves training stability by incorporating target policy smoothing, which introduces noise when selecting target policy actions.

- PPO: PPO is a reinforcement learning method based on policy optimization. The core idea of PPO is to directly optimize the policy function without explicitly learning the value function, in order to maximize cumulative rewards. By controlling the step size of policy parameter updates, PPO ensures that the difference between the new and old policies remains moderate, thereby enhancing training stability and convergence speed.

6.3. Complexity analysis of comparative algorithms

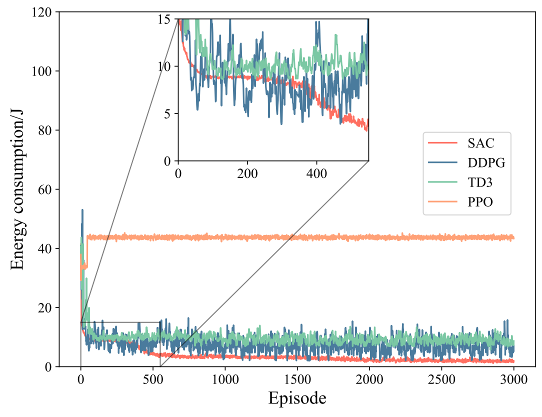
- SAC: In each update step, the SAC algorithm needs to update the policy of actor network and two critics networks, resulting in a higher time complexity, making the per-step computational complexity approximately $O(3\mathcal{N})$, where $\mathcal{N}$ is the size of the neural network. SAC needs to store one actor network, two critic networks, and two target networks, leading to a space complexity of approximately $O(5\mathcal{N})$.

- DDPG: In each update step, DDPG needs to update the policy network and one Q-value network, with a time complexity of approximately $O(2\mathcal{N})$. DDPG requires storing one policy network, one target policy network, one Q-value network, and one target Q-value network, resulting in a space complexity of approximately $O(4\mathcal{N})$.

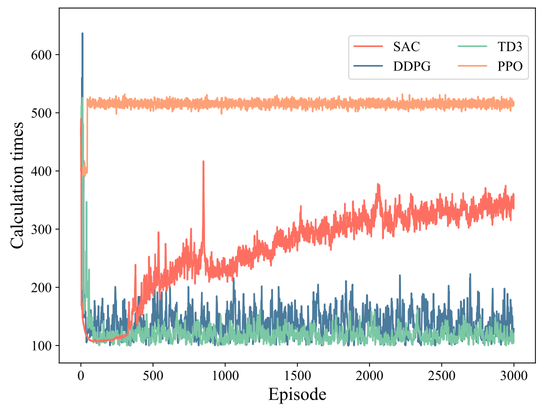
- TD3: In each update step, TD3 needs to update the policy network and two Q-value networks, with a time complexity of approximately $O(3\mathcal{N})$. TD3 requires storing one policy network, one target policy network, two Q-value networks, and two target Q-value networks, leading to a space complexity of approximately $O(6\mathcal{N})$.

- PPO: PPO uses mini batch gradient descent and a clipped probability ratio to stabilize training. The computational complexity per step is related to the neural network size $\mathcal{N}$, the number of optimizer iterations K, and the mini batch size M, resulting in a time complexity of $O(\mathcal{N}KM)$. PPO requires storing a policy network and a value network, leading to a space complexity of approximately $O(2\mathcal{N})$.

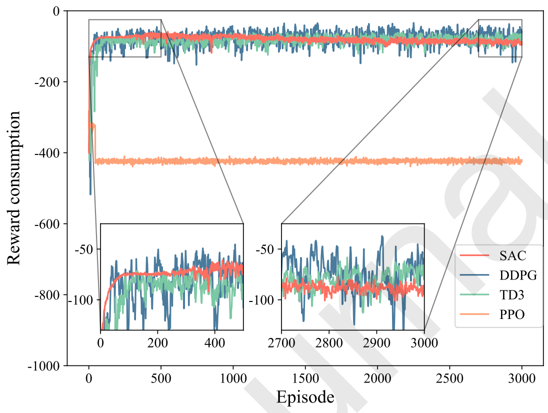
6.4. Simulation results



(a): Energy consumption

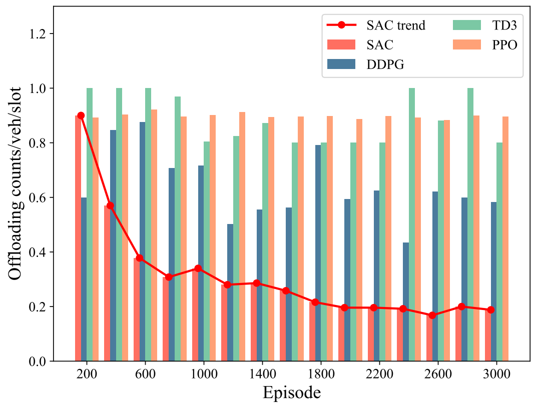


(b): Calculation comparison

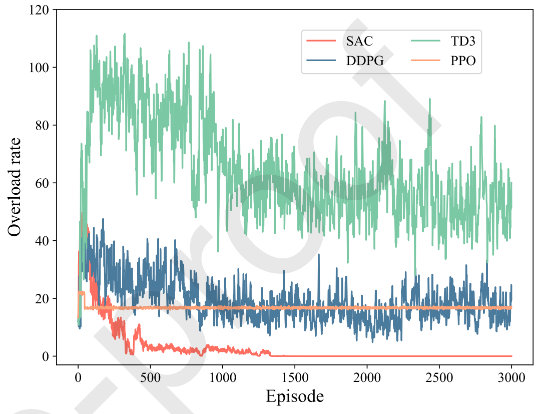


(c): Reward comparison

Fig. 4. Comparison between different DRL algorithms



(a): Offloading counts



(b): Overload rate

Fig. 5. Comparison under different evaluation metrics

In Fig. 4, we show a comparison of different DRL methods with five vehicles taking part in each round of FL. The training methods include SAC, DDPG, TD3, and PPO. We assess these methods in terms of energy consumption, computation times, and reward. Regarding energy consumption, as shown in Fig. 4(a), PPO algorithm fails to effectively reduce energy consumption, with its energy consumption fluctuating around $(43.69 \pm 1.24)J$. In contrast, the energy consumption of the SAC, DDPG, and TD3 algorithms shows a rapid decline and gradually converges to a relatively stable value. Specifically, the SAC algorithm exhibits the fastest convergence speed and is more stable compared to other two algorithms, with the final energy consumption stabilizing at $(2.18 \pm 1.02)J$. The inset provides detailed information for the episodes range $[1, 500]$. It is evident from the inset that TD3 algorithm exhibits less fluctuation than DDPG algorithms but DDPG algorithm can explore lower energy consumption values. Ultimately, the energy consumption of the DDPG algorithm stabilizes at $(8.95 \pm 6.87)J$, while that of the TD3 algorithm stabilizes at $(6.24 \pm 3.54)J$.

As shown in Fig. 4(b), in terms of computation times, all four algorithms eventually converged to a stable value. The results show that the computation times of the SAC algorithm gradually stabilized at 338 ± 36 times, significantly higher than other two algorithms. The final computation times of the DDPG and TD3 algorithms are similar, at 162 ± 61 times and 126 ± 26 times, respectively, but the DDPG algorithm exhibits greater fluctuation. From the figure, we can also observe that PPO algorithm has the highest computation times, stabilizing at 516 ± 14 times. Theoretically, more computations result in a more accurate global model. However, combined with the results of energy consumption shown in Fig. 4(a), PPO algorithm also has the highest energy consumption. This suggests that PPO is not an ideal choice when the goal is to reduce energy consumption. Additionally, Fig. 4(c) shows the reward function of these methods. It is clear that all four algorithms eventually converge to a relatively stable reward value. The inset in the lower left corner provides detailed information for the episode range $[1, 500]$, clearly showing that the reward value of SAC algorithm quickly converged and stabilized after 76 episodes. The DDPG and TD3 algorithms also reach to a relatively stable state after 150 episodes. The inset in the right shows that the final reward values of the SAC, DDPG, and TD3 algorithms stabilized at -88.07 ± 12.79, -91.62 ± 54.56, and -85.53 ± 27.36, respectively. It is also evident from the inset

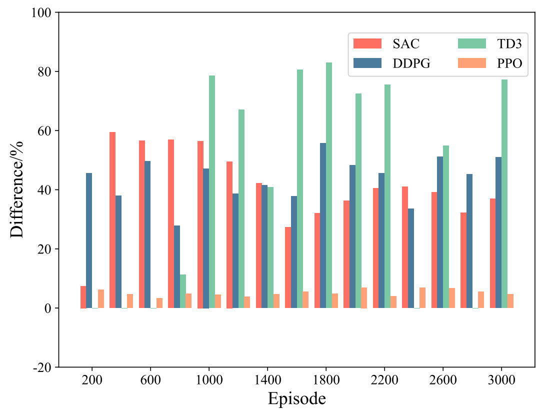


Fig. 6. The difference between with and without offloading threshold

that the reward values of the DDPG and TD3 algorithms tend to surpass that of the SAC algorithm, but in terms of the final value of energy consumption and stability, the SAC algorithm remains the best.

Overall, the SAC algorithm demonstrates the lowest energy consumption. Although its final reward is slightly lower than those of the DDPG and TD3 algorithms, the SAC algorithm converges more rapidly and stably across all the three considered metrics. This effectiveness is primarily due to SAC's ability to balance exploration and exploitation during policy updates, enhancing convergence efficiency. Consequently, SAC proves to be the most effective algorithm for achieving the goal of minimizing energy consumption, making it the best performer overall.

Fig. 5 presents the offloading counts for each vehicle in each slot and overload ratio $\mathcal{R}_0$ under different DRL algorithms. Fig. 5(a) shows that the offloading counts for the DDPG, TD3, and PPO algorithms fluctuate and do not exhibit a convergence trend, whereas the offloading counts for the SAC algorithm demonstrate a stable decline and eventually converge. Fig. 5(b) shows that, except for PPO, the overload ratio $\mathcal{R}_0$ of other three algorithms decreases, with the overload ratio of SAC algorithm approaching zero after the 1332 episodes. From the analysis of Fig. 5(a) and Fig. 5(b), it can be seen that the SAC algorithm gradually reduces the offloading counts during training, favoring more local training on the vehicles, thus reducing the overload ratio and eventually achieving a balance between local vehicle training and offloading to the RSU. Although the DDPG and TD3 algorithms do not significantly reduce the offloading counts during training, they can still lower the overload ratio. This indicates that these two algorithms tend to offload training tasks to the RSU for processing during the initial phase of training. Combined with Fig. 4(a), this also explains why the energy consumption of these two algorithms is higher than that of the SAC algorithm. From Fig. 5, we can also observe that PPO algorithm tends to offload tasks to the RSU for processing, but the number of each offloading tasks do not significantly exceed the total tasks in vehicle side. Combined with Fig. 4, we can see that this policy of PPO algorithm increases transmission costs. Therefore, the PPO algorithm consumes more energy.

Fig. 6 illustrates the differences between offloading efficiency with and without the threshold q_0. From the overall data, it is evident that all values are above 0, indicating that the offloading efficiency with the threshold q_0 is consistently higher than without it. This phenomenon validates that our proposed method of setting the threshold q_0 can significantly enhance offloading efficiency. Specifically, among all algorithms, the TD3 algorithm exhibits the greatest improvement in offloading efficiency, followed by the SAC and DDPG algorithms, with the PPO algorithm showing the least improvement. The superior performance of the TD3 algorithm may stem from its twin delayed deep deterministic policy gradient mechanism, which effectively reduces the variance during the policy network update, thereby enhancing the accuracy of offloading decisions. The SAC algorithm, by introducing the concept of entropy and encouraging exploration diversity, also improves offloading efficiency to a certain extent. In contrast, although the PPO algorithm provides a degree of stability and safety, its relatively conservative policy updates may limit its performance in complex environments.

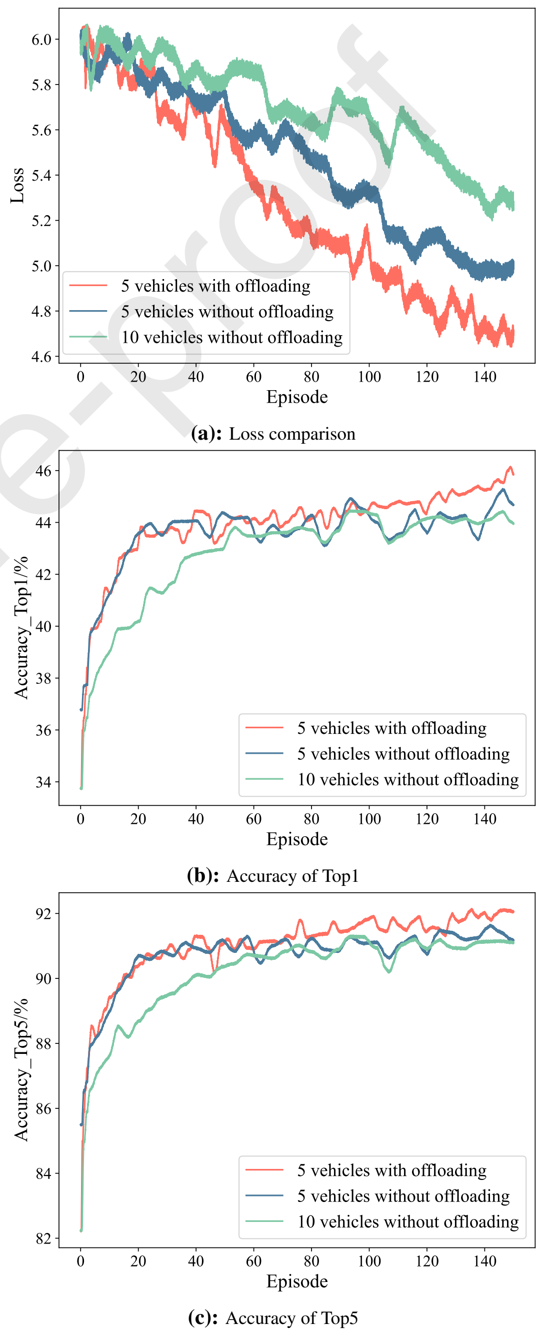
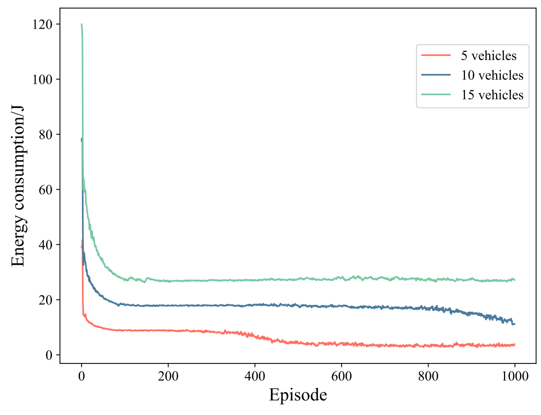
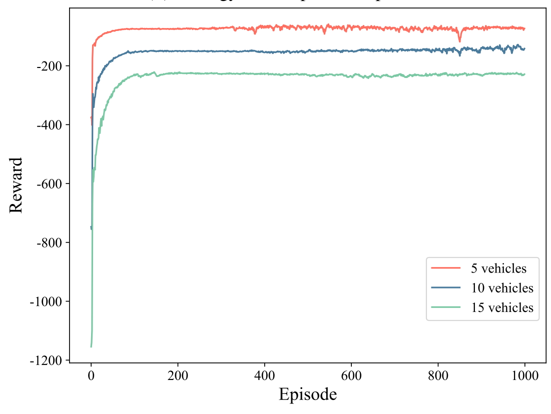


Fig. 7. Comparison of loss function and accuracy. In the case with task offloading, when 5 vehicles participate in federated SSL, in each round of training, the model aggregation involves 10 models; whereas, such as FLSimCo, without task offloading, the aggregation involves only 5 models. Similarly, when 10 vehicles are involved in training without task offloading, the model aggregation includes 10 models.



(a): Energy consumption comparison



(b): Reward function

Fig. 8. Comparison between different numbers of vehicles

In Fig. 7, we compare the trends in the loss function, the Top1 and Top5 classification accuracy of the global model of FLSimCo (without offloading) and our method (with offloading) under varying numbers of vehicles and local iterations per round. It is obvious that the loss function gradually decreases and converges, while the accuracy steadily increases and converges. Specifically, as illustrated in Fig. 7(a), when comparing the cases of 5 vehicles (with 5 models aggregated per round) and 10 vehicles (with 10 models aggregated per round) without task offloading, we observe that with 10 vehicles, the loss function fluctuates more and converges more slowly. This indicates that as the number of vehicles increases, the model diversity also increases, negatively impacting the effectiveness of federated aggregation. However, by offloading tasks to the RSU for training, we obtain models trained by the RSU, which are then sent back to the BS for aggregation along with the locally trained models. Therefore, with 5 vehicles and task offloading, the final number of models is doubled, resulting in 10 local models aggregated per round, which is the same as the number of models aggregated by the BS with 10 vehicles without task offloading. From the figure, we can see that the 5 vehicles with task offloading (10 models in total) not only outperform the 10 vehicles without task offloading but also the 5 vehicles without task offloading. This is because offloading some tasks to the RSU allows the completion of all scheduled iterations, and more local iterations help mitigate the negative impact of model diversity on federated aggregation. Fig. 7(b) and Fig. 7(c) show that in the first 60 episodes, the fewer models aggregated per round, the faster the accuracy improves. However, as the number of episodes increases, this diversity gradually diminishes, resulting in the accuracy of global with 5 and 10 vehicles without task offloading gradually converging. Additionally, when 5 vehicles participate in federated SSL and offload partial iterations task to RSU, the number of models used in the final aggregation is 10. During the first 60 episodes, the accuracy is not affected by model diversity. Instead, it increases rapidly and steadily. In the later stages of training, due to more iterations per round, the accuracy improvement is significantly better than others. This indicates that a greater number of iterations mitigates the negative effects of model diversity during aggregation.

In Fig. 8, we compare the energy consumption and reward obtained by the SAC algorithm with different numbers of vehicles. From Fig. 8(a) and Fig. 8(b), it can be observed that the energy consumption curve shows a decreasing trend and gradually stabilize and the reward function shows a increasing trend and gradually stabilize. As the number of vehicles increases, the episodes required to reach a stable state also grow, reflecting the heightened complexity of task offloading and resource allocation in multi-vehicle scenarios. Overall, with more vehicles participating in training, total system energy consumption rises, while reward values decline. Additionally, these two figures clearly illustrate the SAC algorithm's superiority in convergence speed and stability.

We save the models trained by the SAC, DDPG, and TD3 algorithms from Fig. 4 and conduct test on them. This test includes four rounds, each consisting of 50 independent experiments. The results of each round are obtained by calculating the mean of these 50 experiments. Finally, we further average the means of these four rounds to derive the overall average results of the comprehensive experiments, which are presented in Fig. 9. In the box plot of Fig. 9, we analyze the results from three aspects: energy consumption, computational times, and reward. From the box plot in Fig. 9, it can be seen that the test results of the models trained by the SAC algorithm have a lower degree of dispersion, indicating that the data is concentrated and the fluctuation range is small. In contrast, the test results of the models trained by the DDPG algorithm are relatively dispersed, indicating a certain degree of variability. Specifically, in Fig. 9(a), although there is an outlier in the SAC algorithm, this outlier is relatively close to the lower whisker, and the overall data values are concentrated within a small range, showing a compact distribution with few outliers. Therefore, this outlier can be considered negligible for the overall analysis. However, in Fig. 9(c), both the DDPG and TD3 algorithms have outliers, especially the DDPG algorithm, where the outlier is far from the lower whisker, further indicating the relative instability of these two algorithms. In summary, the models trained by the SAC algorithm demonstrate higher stability and superior performance.

7. Conclusions

In this paper, we proposed an offloading task and resource allocation algorithm based on the SAC algorithm in federated SSL. It employed this algorithm to allocate the transmission powers from vehicles to BS and RSU, CPU frequency for vehicle local training, assignment ratio of RSU computing resource. Based on the assignment ratio and the computing capability of RSU, this algorithm divided the local iterations task of vehicle into two parts: training locally and offloaded to the RSU. We also aimed to minimize the total energy consumption, including local training, RSU training, and transmission. Additionally, we set

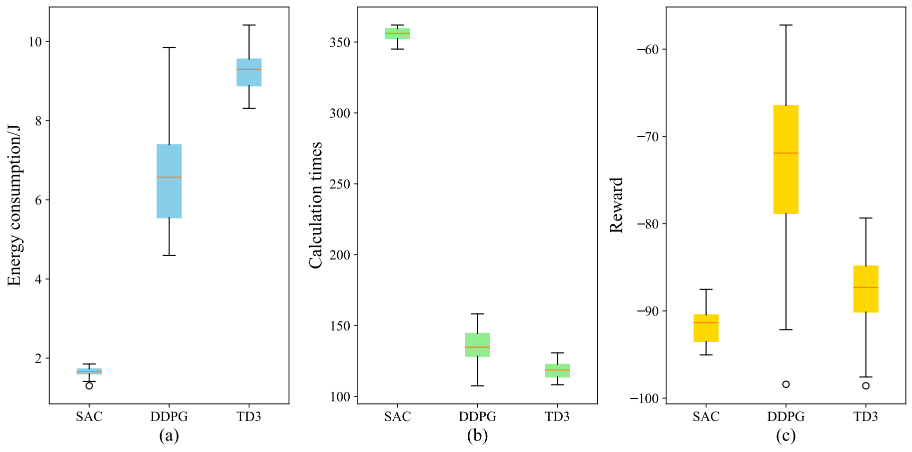


Fig. 9. Test results

an offloading threshold. When the assignment ratio is below this threshold, the vehicle will not perform offloading operation. The conclusions in this paper are as follows

- Our proposed new algorithm is suitable for mobile environments and can support real-time decision-making based on the current environmental state.

- Vehicles will only offload tasks to the RSU when the assignment ratio exceeds the offloading threshold. This design avoids unnecessary offloading operations, thereby reducing transmission overhead and communication resource waste. By setting a reasonable offloading threshold, we can optimize system performance and improve offloading efficiency

- Our proposed algorithm dynamically allocates the number of iterations task between local training and offloading to the RSU, and gradually reduces the offloading frequency during exploration, making full use of the local computing resource. At the same time, it minimizes overload and enhances the utilization of RSU computing resource.

- Our proposed algorithm increased the number of local iterations while minimizing energy consumption. Although the number of models used during aggregation increases, it still improves classification accuracy compared to algorithm that do not offload task, which compensates for the impact of model diversity on the global model during aggregation.

While the approach we suggested has shown promising results in smaller-scale IoV settings, it could encounter challenges related to scalability in larger or more intricate networks. In the future, we will further investigate this aspect to enhance the scalability of the scheme.

References

[1] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, J. Ren, SDN/NFV-empowered future iov with enhanced communication, computing, and caching, Proceedings of the IEEE 108 (2) (2020) 274–291.
[2] R. Zhang, K. Xiong, H. Du, D. Niyato, J. Kang, X. Shen, H. Vincent Poor, Generative ai-enabled vehicular networks: Fundamentals, framework, and case study, IEEE Network 38 (4) (2024) 259 – 267.
[3] K. Zheng, Q. Zheng, P. Chatzimisios, W. Xiang, Y. Zhou, Heterogeneous vehicular networking: A survey on architecture, challenges, and solutions, IEEE Communications Surveys & Tutorials 17 (4) (2015) 2377 – 2396.
[4] R. Deng, Y. Zhang, H. Zhang, B. Di, H. Zhang, H. V. Poor, L. Song, Reconfigurable holographic surfaces for ultra-massive mimo in 6g: Practical design, optimization and implementation, IEEE J. Selected Areas Commun. 41 (8) (2023) 2367 – 2379.
[5] R. Zhang, K. Xiong, Y. Lu, B. Gao, P. Fan, K. B. Letaief, Joint coordinated beamforming and power splitting ratio optimization in mu-miso swipt-enabled hetnets: A multi-agent ddqn-based approach, IEEE J. Selected Areas Commun. 40 (2) (2022) 677–693.
[6] G. Luo, C. Shao, N. Cheng, H. Zhou, H. Zhang, Q. Yuan, J. Li, Edgecooper: Network-aware cooperative lidar perception for enhanced vehicular awareness, IEEE Journal on Selected Areas in Communications 42 (1) (2023) 207–222.
[7] F. Wu, F. Lyu, H. Wu, J. Ren, Y. Zhang, X. Shen, Characterizing user association patterns for optimizing small-cell edge system performance, IEEE Network 73 (3) (2022) 210–217.
[8] J. Zhang, P. Fan, K. B. Letaief, Network coding for efficient multicast routing in wireless ad-hoc networks, IEEE Transactions on Communications 56 (4) (2008) 598 – 607.
[9] Y. Cui, Y. Liang, R. Wang, Intelligent task offloading algorithm for mobile edge computing in vehicular networks, in: 2020 IEEE 91st Vehicular Technology Conference, VTC, 2020.
[10] S. Yue, S. Zeng, L. Liu, Y. C. Eldar, B. Di, Hybrid near-far field channel estimation for holographic mimo communications, in: IEEE Transactions on Wireless Communications, Early Access, 2024.
[11] P. Fan, C. Feng, Y. Wang, N. Ge, Investigation of the time-offset-based qos support with optical burst switching in wdm networks, 2002 IEEE International Conference on Communications. Conference Proceedings-doi:10.1109/ICC.2002.997330.
[12] X. Gu, Q. Wu, P. Fan, Q. Fan, Mobility-aware federated self-supervised learning in vehicular network, Urban Lifeline 2 (10).
[13] J. Shen, N. Cheng, X. Wang, F. Lyu, W. Xu, Z. Liu, K. Aldubaikhy, X. Shen, Ringsfl: An adaptive split federated learning towards taming client heterogeneity, IEEE Transactions on Mobile Computing 23 (5) (2023) 5462–5478.
[14] Y. Liu, J. zhang, Y. zhang, Z. Yuan, G. Liu, A shared cluster-based stochastic channel model for integrated sensing and communication systems, IEEE Transactions on Vehicular Technology 73 (5) (2024) 6032 – 6044.
[15] R. Zhang, K. Xiong, Y. Lu, P. Fan, D. W. K. Ng, K. B. Letaief, Energy efficiency maximization in ris-assisted swipt networks with rsma: A ppo-based approach, IEEE Transactions on Vehicular Technology 41 (5) (2023) 1413–1430.
[16] Z. Wang, Z. Zhong, M. Ni, A semi-markov decision process based computation offloading strategy in vehicular networks, in: 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications, PIMRC, 2018.
[17] L. Yang, H. Zhang, M. Li, J. Guo, H. Ji, Mobile edge computing empowered energy efficient task offloading in 5g, IEEE Transactions on Vehicular Technology 67 (7) (2018) 6398–6409.
[18] M. Zhu, Y. Hou, X. Tao, T. Sui, L. Gao, Joint optimal allocation of wireless resource and mec computation capability in vehicular network, in: 2020 IEEE Wireless Communications and Networking Conference Workshops, WCNCW, 2020.
[19] C. You, K. Huang, H. Chae, B. H. Kim, Energy-efficient resource allocation for mobile-edge computation offloading, IEEE Transactions on Wireless Communications 16 (3) (2016) 1397 – 1411.

[20] W. Wang, N. Cheng, M. Li, T. Yang, C. Zhou, C. Li, F. Chen, Value matters: A novel value of information-based resource scheduling method for cavs, IEEE Transactions on Vehicular Technology 73 (6) (2024) 8720–8735.

[21] D. Deng, X. Li, V. Menon, M. J. Piran, H. Chen, M. A. Jan, Learning based joint uav trajectory and power allocation optimization for secure iot networks, Digital Communications and Networks 8 (4) (2022) 415–421.

[22] Q. Wu, W. Wang, P. Fan, Q. Fan, J. Wang, K. B. Letaief, Urllc-awared resource allocation for heterogeneous vehicular edge computing, IEEE Transactions on Vehicular Technology 73 (8) (2024) 11789 – 11805.

[23] X. Wang, Q. Wu, P. Fan, Q. Fan, H. Zhu, J. Wang, Vehicle selection for c-v2x mode 4 based federated edge learning systems, IEEE Systems Journal (2024) 1–12.

[24] Q. Wu, W. Wang, P. Fan, Q. Fan, H. Zhu, K. B. Letaief, Cooperative edge caching based on elastic federated and multi-agent deep reinforcement learning in next-generation networks, IEEE Transactions on Network and Service Management 21 (4) (2024) 4179 – 4196.

[25] N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, X. Shen, Space/aerial-assisted computing offloading for iot applications: A learning-based approach, IEEE Journal on Selected Areas in Communications 37 (5) (2019) 1117–1129.

[26] R. Sun, N. Cheng, C. Li, F. Chen, W. Chen, Knowledge-driven deep learning paradigms for wireless network optimization in 6g, IEEE Network 38 (2) (2024) 70–78.

[27] J. Xu, X. Liu, X. Zhu, Deep reinforcement learning based computing offloading and resource allocation algorithm for mobile edge networks, in: 2020 IEEE 6th International Conference on Computer and Communications, ICCC, 2020.

[28] J. Yu, C. Ma, B. Yang, Y. Wei, Blockchain-enabled rcs task offloading and resource allocation policy using drl approach, in: 2022 IEEE 10th International Conference on Computer Science and Network Technology, ICCSNT, 2022.

[29] S. Chen, Q. Wang, J. Chen, T. Wu, An intelligent task offloading algorithm (itoa) for uav network, in: 2019 IEEE Globecom Workshops, GC Wkshps, 2020.

[30] B. Wang, L. Liu, J. Wang, Actor-critic based drl algorithm for task offloading performance optimization in vehicle edge computing (2023).

[31] H. Cui, D. Wang, Q. Li, X. Liu, H. Lu, A2c deep reinforcement learning-based mec network for offloading and resource allocation, in: 2021 7th International Conference on Computer and Communications, ICCC, 2022.

[32] W. H. Chen, Y. Ren, J. C. Chen, Design and analysis of a threshold offloading (to) algorithm for lte femtocell/macrocell networks, in: 2016 IEEE Symposium on Computers and Communication, ISCC, 2016.

[33] O. M. Al-Tuhafi, E. H. Al-Hemiary, Adaptive thresholds for task offloading in iot-edge-cloud networks, in: 2023 International Conference On Cyber Management And Engineering, CyMaEn, 2023.

[34] X. Qin, B. Li, L. Ying, Efficient distributed threshold-based offloading for large-scale mobile cloud computing, IEEE/ACM Transactions on Networking 31 (1) (2020) 308–321.

[35] N. Jahan, G. Bajwa, T. Akilan, Federated learning-assisted self-supervised cnn for monkeypox diagnosis, in: 2023 IEEE Western New York Image and Signal Processing Workshop, WNYISPW, 2023.

[36] M. Feng, C. C. Kao, Q. Tang, M. Sun, V. Rozgic, S. Matsoukas, C. Wang, Federated self-supervised learning for acoustic event classification, in: ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2022.

[37] S. Wei, G. Cao, C. Dai, S. Dai, B. Guo, Fedco: Self-supervised learning in federated learning with momentum contrast, in: 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application, HPCC/DSS/SmartCity/DependSys, 2023.

[38] S. Wu, N. Chen, G. Wen, L. Xu, P. Zhang, H. Zhu, Virtual network embedding for task offloading in iiot: A drl-assisted federated learning scheme, IEEE Transactions on Industrial Informatics 20 (4) (2024) 6814–6824.

[39] L. Zang, X. Zhang, B. Guo, Federated deep reinforcement learning for online task offloading and resource allocation in wpc-mec networks, IEEE Access 10 (2022) 9856–9867.

[40] T. Zhao, F. Li, L. He, Secure video offloading in mec-enabled iiot networks: A multicell federated deep reinforcement learning approach, IEEE Transactions on Industrial Informatics 20 (2) (2023) 1618– 1629.

[41] M. Ji, Q. Wu, N. Cheng, W. Chen, J. Wang, K. B. Letaief, Graph neural networks and deep reinforcement learning based resource allocation for v2x communications, IEEE Internet of Things Journal.

[42] K. Qi, Q. Wu, P. Fan, N. Cheng, W. Chen, K. B. Letaief, Reconfigurable intelligent surface aided vehicular edge computing: Joint phase-shift optimization and multi-user power allocation, IEEE Internet of Things Journal.

[43] Z. Shao, Q. Wu, P. Fan, N. Cheng, W. Chen, J. Wang, K. B. Letaief, Semantic-aware spectrum sharing in internet of vehicles based on deep reinforcement learning, IEEE Internet of Things Journal.

[44] C. Zhang, W. Zhang, Q. Wu, P. Fan, Q. Fan, J. Wang, K. B. Letaief, Distributed deep reinforcement learning based gradient quantization for federated learning enabled vehicle edge computing, IEEE Internet of Things Journal.

[45] K. Qi, Q. Wu, P. Fan, N. Cheng, W. Chen, J. Wang, K. B. Letaief, Deep-reinforcement-learning-based aoi-aware resource allocation for ris-aided iov networks, IEEE Transactions on Vehicular Technology.

[46] H. Xiao, J. Zhao, Q. Pei, J. Feng, L. Liu, W. Shi, Vehicle selection and resource optimization for federated learning in vehicular edge computing, IEEE Transactions on Intelligent Transportation Systems 23 (8) (2021) 11073–11087.

[47] Q. Wu, Y. Zhao, Q. Fan, P. Fan, J. Wang, C. Zhang, Mobility-aware cooperative caching in vehicular edge computing based on asynchronous federated and deep reinforcement learning, IEEE Journal of Selected Topics in Signal Processing 17 (1) (2022) 66–81.

[48] K. Qi, Q. Wu, P. Fan, N. Cheng, Q. Fan, J. Wang, Reconfigurable intelligent surface assisted vec based on multi-agent reinforcement learning, IEEE Communications Letters 28 (10) (2024) 2427–2431.

[49] Z. Shao, Q. Wu, P. Fan, N. Cheng, Q. Fan, J. Wang, Semantic-aware resource allocation based on deep reinforcement learning for 5g-v2x hetnets, IEEE Communications Letters 28 (10) (2024) 2452–2456.

[50] C. Zhang, X. Xu, Q. Wu, P. Fan, Q. Fan, H. Zhu, J. Wang, Anti-byzantine attacks enabled vehicle selection for asynchronous federated learning in vehicular edge computing, China Communication 73 (8) (2024) 1–17.

[51] S. Luo, X. Chen, Q. Wu, Z. Zhou, S. Yu, Hfel: Joint edge association and resource allocation for cost-efficient, IEEE Transactions on Wireless Communications 19 (10) (2020) 6535–6548.

[52] Q. Wu, S. Wang, H. Ge, P. Fan, Q. Fan, K. B. Letaief, Delay-sensitive task offloading in vehicular fog computing-assisted platoons, IEEE Transactions on Network and Service Management 21 (2) (2023) 2012 – 2026.

[53] A. v. d. Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding. `https://arxiv.org/abs/1807.03748`, 2019 (accessed 22 january 2019).

[54] K. Hou, X. Luv, W. Zhang, An adaptive fusion panoramic image mosaic algorithm based on circular lbp feature and hsv color system, in: 2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence, ICIBA, 2020.

[55] C. Zhang, K. Zhang, T. X. Pham, A. Niu, Z. Qiao, C. D. Yoo, I. S. Kweon, Dual temperature helps contrastive learning without many negative samples: towards understanding and simplifying moco. `https://arxiv.org/abs/2203.17248`, 2022 (accessed 30 march 2022).

[56] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actorcritic: Off-policy maximum entropy deep reinforcement learningwith a stochastic actor. `https://arxiv.org/abs/1801.01290`, 2018 (accessed 8 august 2018).

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: