# Collective Deep Reinforcement Learning for Intelligence Sharing in the Internet of Intelligence-Empowered Edge Computing

Qinqin Tang, Renchao Xie, *Senior Member, IEEE*, Fei Richard Yu, *Fellow, IEEE*, Tianjiao Chen, Ran Zhang, Tao Huang, *Senior Member, IEEE*, and Yunjie Liu

**Abstract**—Edge intelligence is emerging as a new interdiscipline to push learning intelligence from remote centers to the edge of the network. However, with its widespread deployment, new challenges arise in terms of training efficiency and service of quality (QoS). Massive repetitive model training is ubiquitous due to the inevitable needs of users for the same types of data and training results. Additionally, a smaller volume of data samples will cause the over-fitting of models. To address these issues, driven by the Internet of intelligence, this article proposes a distributed edge intelligence sharing scheme, which allows distributed edge nodes to quickly and economically improve learning performance by sharing their learned intelligence. Considering the time-varying edge network states including data collection states, computing and communication states, and node reputation states, the distributed intelligence sharing is formulated as a multi-agent Markov decision process (MDP). Then, a novel collective deep reinforcement learning (CDRL) algorithm is designed to obtain the optimal intelligence sharing policy, which consists of local soft actor-critic (SAC) learning at each edge node and collective learning between different edge nodes. Simulation results indicate our proposal outperforms the benchmark schemes in terms of learning efficiency and intelligence sharing efficiency.

**Index Terms**—Distributed intelligence sharing, Internet of intelligence, edge computing, collective deep reinforcement learning

## 1 Introduction

In recent years, the rapid growth of the mobile Internet and the Internet of things (IoT) has triggered an explosive growth of data traffic on the edge side. Data processing through cloud computing will lead to colossal bandwidth consumption during the transmission process and impose heavy computing pressure on the cloud [1]. In addition, the high latency of cloud computing is not suitable for tasks that require real-time response. Driven by this trend, computing power is shifting from the centralized cloud to the edge [2], [3]. Meanwhile, the vigorous development of artificial intelligence (AI) makes it possible to quickly perceive and train massive amounts of local data, which can not only adapt to the ever-changing environment but also significantly reduce latency and improve computing efficiency [4]. Accordingly, edge intelligence, which integrates edge computing and AI, will undoubtedly become a powerful driver for future network development [5], [6], [7], [8].

However, the widespread deployment of edge intelligence also brings new challenges [9]. In edge networks, users will inevitably generate similar machine learning (ML) tasks, which may require the same types of data and even expect the same training results. Massive repetitive model training has led to a large number of redundant calculations in the network and a severe waste of limited edge resources. Additionally, a smaller volume of data samples will also cause the over-fitting of models. Therefore, invalid and meaningless model training prevails, resulting in low training efficiency and reduced quality of service (QoS).

Nowadays, there have been various prior studies investigating how to effectively use the limited resources of edge networks to further improve the efficiency or QoS of edge intelligence, such as data sharing [10], [11], gradient compression [12], [13], adaptive global aggregation [14], and so forth. Data sharing improves the QoS of edge models, but it needs to calculate a large number of data samples, and high transmission costs will also be generated to transmit the shared data. Gradient compression and adaptive global aggregation can effectively ensure learning efficiency but at the expense of

- *Qinqin Tang is with the Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen University, Shenzhen, Guangdong 518107, China. E-mail: qqtang@bupt.edu.cn.*
- *Renchao Xie, Ran Zhang, Tao Huang, and Yunjie Liu are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the Purple Mountain Laboratories, Nanjing 211111, China. E-mail: {Renchao_xie, zhangran, htao}@bupt.edu.cn, liuyj@chinaunicom.cn.*
- *Fei Richard Yu is with the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen University, Shenzhen, Guangdong 518107, China, and also with the School of Information Technology, Carleton University, Ottawa, ON K1S 5B6, Canada. E-mail: richard.yu@carleton.ca.*
- *Tianjiao Chen is with the China Mobile Research Institute, Beijing 100053, China. E-mail: jiaotianchen@163.com.*

*Manuscript received 29 November 2021; revised 5 June 2022; accepted 15 August 2022. Date of publication 18 August 2022; date of current version 3 October 2023.*
*This work was supported in part by the National Natural Science Foundation of China under Grant 62171046, in part by the Natural Science Foundation of Beijing under Grants 4212004 and L201002, in part by the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ) under Grant GML-KF-22-24, and in part by the China Postdoctoral Science Foundation under Grant 2022TQ0048.*
*(Corresponding author: Renchao Xie.)*
*Digital Object Identifier no. 10.1109/TMC.2022.3199812*

QoS. There is an imminent need for innovative schemes to joint optimize these two metrics. Fortunately, recent advances in the Internet of intelligence have provided an alternative solution [15], [16], [17]. Intelligence is a further abstraction and concentration of information (data). Internet of intelligence is conceived as an emerging networking paradigm through intelligence networking, making intelligence as easily to obtain as information (data). Empowering edge computing with the Internet of intelligence, distributed edge intelligence can be stored, transmitted and shared through networking, thereby reducing information redundancy in the network and significantly improving training efficiency and model QoS.

Therefore, focusing on the Internet of intelligence-empowered edge computing, this paper aims to explore an efficient intelligence sharing scheme for distributed edge nodes to share their learned intelligence. By aggregating the collected intelligence, edge nodes can quickly improve the performance of local models without spending a lot of computing resources in calculating massive data samples. Cooperative model training has been studied in some previous research works, such as the client selection in federated learning (FL) [18], [19], [20]. In such works, a central server selects the clients participating in the model training and obtains a global model by aggregating the model parameters from the clients to optimize the performance of the trained global model. Different from these works, in the proposed intelligence sharing problem, each edge node is both a client and a server, which can train intelligence locally as well as directly request intelligence from other edge nodes. Our objective is to optimize the utility of intelligence model training, which includes not only the performance gain of the trained model, but also the revenue/cost of sharing/requesting intelligence and the energy cost of training/transmitting intelligence. Therefore, in addition to the selection of edge nodes participating in intelligence sharing, considering the interactions between decisions made by different edge nodes, the training level of local intelligence models and the resource allocation of edge computing networks are also critical, which have not been well studied in the current client selection problem. Specifically, our major contributions are summarized as follows.

- We propose a distributed intelligence sharing model for the Internet of intelligence-empowered edge computing. Due to the difficulty of managing heterogeneous edge nodes and the potential laziness and dishonest behavior of edge servers, reputation is introduced as a fairness metric to select reliable edge nodes for intelligence sharing to resist unreliable sharing operations in the edge network.
- Based on the proposed model, we formulate the distributed intelligence sharing problem as a multi-agent Markov decision process (MDP), taking into account the volume of collected data, the quality level of collected data, the computing capability of edge nodes, the channel condition of edge networks and the reputation opinions between edge nodes. Our objective is to optimize the overall utility of edge nodes through intelligence sharing, which is a trade-off between the accuracy of trained models, the revenue/cost of intelligence sharing, and the energy cost.
- To solve the multi-agent MDP formulation, inspired by the idea of "collective learning", a collective deep reinforcement learning (CDRL) algorithm is designed to obtain the optimal intelligence sharing policies, including training iteration selection, intelligence requesting, and spectrum resource allocation. Specifically, a DRL algorithm based on soft actor-critic (SAC) is proposed for local learning at each edge node. Then, the CDRL is used between different edge nodes to further accelerate learning and avoid local optimum.
- The performance of our proposed intelligence sharing scheme is evaluated through extensive simulations. Numerical results validate the effectiveness and superiority of our proposal compared with the benchmark schemes in terms of learning efficiency and intelligence sharing efficiency.

The remainder rest of this paper is organized as follows. We first present the related work in Section 2. Then, we describe the Internet of intelligence-empowered edge computing system in Section 3. We formulate the distributed intelligence sharing problem as a multi-agent MDP in Section 4. To solve the proposed multi-agent MDP formulation, we develop a CDRL-based intelligence sharing algorithm to find the approximate optimal intelligence sharing policies in Section 5. Finally, we demonstrate the numerical results in Section 6 and conclude this paper in Section 7.

## 2 Related Work

Edge intelligence aims to push learning intelligence from one or several remote centers to the edge of the network. By overcoming the high monetary costs and high transmission delays that may occur when moving large amounts of data through a wide area network (WAN), it is outstanding in terms of reduced bandwidth, time efficiency, energy-saving, and privacy protection. Thanks to such benefits, edge intelligence has attracted widespread attention from industry and academia. In [5], Yang et al. developed a two-tier edge intelligence enabling architecture for autonomous driving and introduced two functional designs to effectively make offloading decisions to ensure data privacy and improve reasoning accuracy while meeting delay constraints. In [6], Zhang et al. demonstrated an edge intelligent and secure framework for the industrial Internet of things (IIoT), and developed a DRL-based edge resource scheduling scheme and a blockchain-enabled credit differentiation transaction approval mechanism to improve service capabilities. Yuan et al. in [7] proposed ICANE, an inductive content-enhanced network embedding model to maintain higher-order structure and semantic content proximity in large-scale decentralized edge networks leveraged by AI. By clustering and classifying HTTP traffic running sequence, An et al. in [8] proposed a new anomaly detection framework based on the latest development of edge intelligence to effectively discover unknown network intrusions. Zhang et al. designed a reinforcement on federated (RoF) scheme that is executed decentralizedly at edge servers to efficiently optimize the performance of FL by collaboratively making optimal device selection and resource allocation decisions.

Due to repetitive training and stochastic data samples, edge intelligence faces challenges in training efficiency and

QoS. As a result, many efforts have been made to improve the efficiency or QoS of edge intelligence. To increase the training speed, Wang et al. in [14] proposed an adaptive global aggregation scheme to control the number of local and global iterations during the training of distributed ML models. In [12], Sattler et al. developed a gradient compression method to accelerate the learning speed and smoothly adapt to the communication constraints in the learning environment, such as computation time, network delay and bandwidth, and temporal inhomogeneities therein. In [13], Wen et al. proposed TernGrad, which uses ternary gradients and gradient clipping to accelerate distributed deep learning in data parallelism. To improve the QoS of edge intelligence, data sharing is considered an effective approach. In [10], Zhou et al. designed a highly flexible data sharing mechanism for distributed ML to improve training quality at a reduced economic cost. Based on FL and deep Q-network, Li et al. in [11] proposed an intelligent collaborative data sharing method for vehicular edge networks to broadcast data in the heterogeneous edge computing layer and effectively adapt to the dynamic vehicular connection environment. However, the majority of the above work either focuses on efficiency improvement or on model QoS guarantee without considering the joint optimization of these two metrics.

Therefore, the Internet of intelligence is emerged to cope with these issues by allowing the networking of distributed intelligence. The concept of the Internet of intelligence is first proposed in [16]. Yu et al. in this work briefly introduced the motivations and challenges of intelligence networking, and two cases of scenarios of the Internet of intelligence are also presented. A novel wireless virtual reality (VR) framework is presented by Lin et al. in [17] for beyond fifth-generation (B5G) systems and the Internet of intelligence. The proposed framework can make full use of intelligence to coordinate computing, storage and transmission systems to realize the ubiquitous deployment of wireless VR. Inspired by the idea of collective learning, Fu et al. developed a blockchain collective learning framework in [21] to support the large-scale autonomous vehicles (CAVs) in the Internet of intelligence and improve the accuracy and security of the learning model. Driven by the Internet of intelligence, this paper therefore considers the efficient intelligence sharing for edge computing networks, which is a field that has not yet been involved in pioneer works. Considering the varying conditions of edge networks and the reliability of edge nodes, our work jointly optimizes model training, intelligence sharing and resource allocation, brings desirable performance improvements and economic benefits.

## 3 System Description

In this section, we first provide an overview of the Internet of intelligence-empowered edge computing model considered in our study. Then, the system assumptions and definitions are presented. The definitions of the main notations used in this paper are listed in Table 1.

### 3.1 System Model

As illustrated in Fig. 1, we consider an Internet of intelligence-empowered edge computing network in which a set of edge nodes denoted by $\mathcal{I} = \{1, 2, \ldots, I\}$ are running an AI application as exemplified by video surveillance, autonomous driving, and IIoT. This AI application continuously senses a large number of multimodal data samples, such as pictures, audios, and videos, from a wide range of smart devices. The quality of AI models (for example, deep neural networks (DNN)) is closely related to the quantity and quality of data samples used for training [10]. Due to the distributed and heterogeneous nature of edge nodes, the quantity and quality of data collected by each edge node is stochastic over time. When the data collected by an edge node is abundant and of high quality, the performance of the trained model is superior. On the contrary, when the quantity/quality of data samples collected by an edge node is small/poor, the performance of the trained model is relatively poor. In such cases, the idea of data sharing may be able to improve the quality of trained AI models. However, data transactions will bring tremendous pressure to the data privacy and security of edge nodes. Additionally, the transmission of large amounts of data will also lead to increased network traffic and transmission costs.

TABLE 1
List of Notations

| Symbol | Description |
|---|---|
| $t$ | Time slot index |
| $\mathcal{I}$ | Set of edge nodes |
| $\Lambda_i$ | Volume of data collected by edge node $i$ |
| $\Upsilon_i$ | Quality level of data collected by edge node $i$ |
| $\Omega_{i,j}$ | Spectrum efficiency for the communication link through edge node $i$ to edge node $j$ |
| $F_i$ | Loss function of the local data set at edge node $i$ |
| $\alpha$ | Temperature parameter |
| $p_{i,j}$ | Transmission power of edge node $i$ for edge node $j$ |
| $\zeta_{i,j}$ | Data correlation between edge node $i$ and $j$ |
| $g_{i,j}$ | Channel gain between edge node $i$ and edge node $j$ |
| $\eta_{i,j}$ | Percentage of the spectrum allocated to edge node $j$ by edge node $i$ |
| $c_{i,j}$ | Transmission rate from edge node $i$ to edge node $j$ |
| $\sigma^2$ | Noise power |
| $W$ | Bandwidth of available spectrum between edge nodes |
| $\varphi_i$ | Model accuracy of edge node $i$ |
| $\tau_i$ | Number of training iterations determined by edge node $i$ |
| $d$ | Indicator of positive events |
| $q$ | Indicator of negative events |
| $\Phi_i$ | Available computing capability of edge node $i$ |
| $R_{i,j}$ | Reputation value of edge node $i$ for edge node $j$ |
| $\iota$ | Weight parameter representing the impact of uncertainty on reputation |
| $d(q)$ | Number of positive (negative) events |
| $\varpi$ | Number of required CPU cycles to execute a data sample |
| $b_{i,j}$ | Indicator of intelligence sharing willingness |
| $EC_i$ | Energy consumption of edge node $i$ for local training |
| $ET_i$ | Energy consumption of edge node $i$ for intelligence requesting |
| $TH_i$ | Compensation threshold of edge node $i$ |
| $r_{i,j}$ | Indicator of intelligence requesting |
| $\varrho$ | Data size of a local model |
| $\epsilon_i$ | Model performance of the aggregated model of edge node $i$ |
| $\rho$ | Performance gain coefficient |
| $\kappa$ | Price coefficient of the unit accuracy of intelligence |

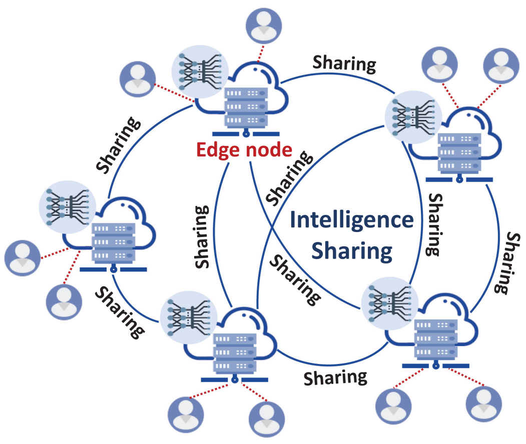


Fig. 1. Illustration of intelligence sharing in the Internet of intelligence-empowered edge computing.

Therefore, direct intelligence sharing between edge nodes becomes an effective way. After each edge node uses its local data samples to train its local models, it can share its learned intelligence with other edge nodes through intelligence sharing. This way, edge nodes can quickly learn models with better performance without sampling and training large amounts of data samples. The privacy and security of edge nodes can also be guaranteed.

Similar to the recent modeling work on edge computing [22], [23], [24], we use a discrete time-slotted model to characterize the system dynamics, e.g., time-varying data sensing and channel condition. The time slot index is denoted by $t$, with $t \in \{0, 1, \ldots, T\}$. The data samples are collected on an online and periodically basis. We define a variable $\Lambda_i(t)$ to denote the quantity of data samples collected by edge node $i$ at time slot $t$. For each edge node $i$, the peak level of the collected data samples is supposed to be $\Lambda_i^{\max}$, such that $\{\Lambda_i(t) \leq \Lambda_i^{\max}, \forall i \in \mathcal{I}, \forall t\}$. The data sensing in the edge computing environment is usually time-varying and unpredictable, e.g., the interested images captured by a video camera may fluctuate over time. Hence, we do not assume that the statistics of $\Lambda_i(t), \forall i \in \mathcal{I}, \forall t$ and $\Lambda_i^{\max}$ have any prior knowledge in our model. Moreover, the quality level of data samples collected by edge nodes in different time slots varies. In time slot $t$, we denote $\Upsilon_i(t)$ as the quality level of data samples collected by edge node $i$. In the considered edge computing network, the data collected between different edge nodes are somewhat related rather than independent of each other. Therefore, we use $\zeta_{i,j} \in [0,1]$ to represent the data correlation between edge node $i$ and edge node $j$.

## 3.2 Communication Model

The information exchange between edge nodes is facilitated through wireless communication. The wireless channels are assumed to be independent and identically distributed (i.i. d) block fading, i.e., the channel varies in different time slots, but remains static within each time slot. Then, according to [25], the spectrum efficiency for the communication link through edge node $i$ to edge node $j$ is

$$\Omega_{i,j}(t) = \log_2\left(1 + \frac{p_{i,j}(t) g_{i,j}(t)}{\sum_{i' \in \mathcal{I}/\{i\}} p_{i',j}(t) g_{i',j}(t) + \sigma^2}\right), \tag{1}$$

where $\sigma^2$ is the noise power. $p_{i,j}(t)$ and $p_{i',j}(t)$ are the transmission power of edge node $i$ and edge node $i'$ at time slot $t$, respectively. $g_{i,j}(t)$ and $g_{i',j}(t)$ are the channel gain of wireless propagation channel between edge node $i$ and edge node $j$, edge node $i'$ and edge node $j$, respectively. We denote $\eta_{i,j}(t) \in [0,1]$ as the percentage of the spectrum allocated to edge node $j$ by edge node $i$ at time slot $t$. Then, the transmission rate from edge node $i$ to edge node $j$ can be calculated as

$$c_{i,j}(t) = \eta_{i,j}(t) W \Omega_{i,j}(t), \tag{2}$$

where $W$ is the bandwidth of available spectrum between edge nodes.

## 3.3 Intelligence Training Model

In typical ML settings, for a data sample $\{\boldsymbol{x}_n, y_n\}$ with a multi-dimension input feature $\boldsymbol{x}_n$, the goal is to find a model parameter vector $\boldsymbol{\omega}$ that characterizes the labeled output $y_n$ with a loss function $f_n(\boldsymbol{\omega})$. Some examples of the loss function include $f_n(\boldsymbol{\omega}) = (1/2)(\boldsymbol{x}_n^\top - y_n)^2$, $y_n \in \mathbb{R}$ for linear regression, and $f_n(\boldsymbol{\omega}) = -\log(1 + \exp(-y_n \boldsymbol{x}_n^\top \boldsymbol{\omega}))$, $y_n \in \{-1, 1\}$ for logistic regression. In more advanced models, such as neural networks or conditional random fields, although the loss function may be more complex and even non-convex, it can still be written as $f_n(\boldsymbol{\omega})$ and use the classical backpropagation method to calculate the gradient efficiently.

For a local data set $i \in \mathcal{I}$ with a number of $D_i$ data samples at edge node $i$, the loss function is defined as

$$F_i(\boldsymbol{\omega}) = \frac{1}{D_i} \sum_{n \in D_i} f_n(\boldsymbol{\omega}) + \xi h(\boldsymbol{\omega}), \tag{3}$$

where $h(\cdot)$ is a regularizer function and can be given as $h(\cdot) = \frac{1}{2}\|\cdot\|^2; \forall \xi \in [0,1]$.

Each edge node $i \in \mathcal{I}$ trains its model in an iterative manner to solve its local problems [26]:

$$\boldsymbol{\omega}_i^* = \arg\min_{\boldsymbol{\omega}} F_i(\boldsymbol{\omega} | \boldsymbol{\omega}_i, \triangledown F_i(\boldsymbol{\omega})), \tag{4}$$

where $\boldsymbol{\omega}_i^*$ is the optimal model parameter for edge node $i$.

In time slot $t$, when $\Lambda_i(t)$ data samples are collected by edge node $i$, it uses these data samples for model training. The accuracy of the model is used in this work to measure the performance of the trained AI model. We use $\varphi \in [0,1]$ to represent the accuracy of the trained model. Model accuracy is closely related to the data used for training, the number of training iterations, the computing capability of computing nodes, the algorithm used for training, and so forth. Let $\tau_i(t)$ denotes the number of training iterations determined by edge node $i$ at time slot $t$. Similar to [27], [28], [29], the model performance of edge node $i$ at time slot $t$ then satisfies

$$\varphi_i(t) = 1 - \exp\big(-\varsigma^{\mathrm{lc}} \Phi_i(t) (\Upsilon_i(t) \Lambda_i(t) \tau_i(t)^{\alpha})^{\upsilon}\big), \tag{5}$$

where $\varsigma^{\mathrm{lc}}$ and $\upsilon$ are weight factors. $\Phi_i(t)$ is the available computing capability of edge node $i$ at time slot $t$. $\alpha$ is the learning factor that reflects the marginal revenue of iterations and depends on the selected learning algorithm.

To characterize the quality of the solution, the local $\varphi_i$ accuracy also satisfies

$$\left\|F_i(\boldsymbol{\omega}_i^*)\right\| \leq (1-\varphi_i)\|F_i(\mathbf{0})\|. \tag{6}$$

Given the two extreme examples, the realization of $\varphi_i = 1$ requires finding the exact maximum, while $\varphi_i = 0$ means that the edge node has not achieved any improvement at all.

## 3.4 Reputation Model

In the considered edge computing network, edge nodes may share false information due to sensor failure, computer virus infection and even selfish purposes. In addition, they may also provide irrelevant information (useless information) to each other. To prevent this, we consider a reputation model to quantify the reputation of edge nodes. With reference to [30], [31], [32], the subjective logic model is adopted to formulate an individual reputation assessment according to the interaction history. In order to obtain a more accurate reputation, each edge node combines its local reputation opinions with reputation opinions recommended by other edge nodes in the same edge network to generate comprehensive reputation values.

### 3.4.1 Local Reputation

The local opinion of edge node $i$ to edge node $j$ in subjective logic can be described as: $\Gamma_{i,j}^{\text{loc}} := \{RB_{i,j}^{\text{loc}}, RD_{i,j}^{\text{loc}}, RU_{i,j}^{\text{loc}}\}$, where $RB_{i,j}^{\text{loc}}$, $RD_{i,j}^{\text{loc}}$, $RU_{i,j}^{\text{loc}}$ represent the belief, disbelief, and uncertainty, respectively. We have $RB_{i,j}^{\text{loc}}, RD_{i,j}^{\text{loc}}, RU_{i,j}^{\text{loc}} \in [0,1]$ and $RB_{i,j}^{\text{loc}} + RD_{i,j}^{\text{loc}} + RU_{i,j}^{\text{loc}} = 1$. Denote $d(t) \in \{0,1\}$ and $q(t) \in \{0,1\}$ to indicate whether there is a positive event or a negative event at time slot $t$, respectively. That is, $d(t) = 1$ ($q(t) = 1$) indicates there is a positive (negative) event at time slot $t$, and vice versa, no interaction events have occurred. Then, the local opinion at time slot $t$ can be given by

$$\begin{cases} RB_{i,j}^{\text{loc}}(t) = \left(1 - RU_{i,j}^{\text{loc}}(t)\right)\frac{\tilde{d}_{i,j}(t)}{\tilde{d}_{i,j}(t)+\tilde{q}_{i,j}(t)} \\ RD_{i,j}^{\text{loc}}(t) = \left(1 - RU_{i,j}^{\text{loc}}(t)\right)\frac{\tilde{q}_{i,j}(t)}{\tilde{d}_{i,j}(t)+\tilde{q}_{i,j}(t)} \\ RU_{i,j}^{\text{loc}}(t) = 1 - oc_{i,j}(t) \end{cases} \tag{7}$$

where $\tilde{d}_{i,j}(t) = \sum_{t'=0}^{t-1} d_{i,j}(t')$ and $\tilde{q}_{i,j}(t) = \sum_{t'=0}^{t-1} q_{i,j}(t')$ are the number of positive events and negative events accumulated in the history before time slot $t$. If edge node $i$ believes that the intelligence provided by edge node $j$ is useful, trusted and reliable for its model update, edge node $i$ considers this interaction to be a positive event between it and edge node $j$, and vice versa. The uncertainty of reputation opinion represents the probability that edge node $i$ is not sure that the intelligence shared by edge node $j$ is true. Due to the time-varying nature of edge computing networks, the communication quality between edge nodes is unstable [33]. When the communication quality is better, the uncertainty is smaller, and when the communication quality is worse, the uncertainty is larger. Therefore, in our study, the uncertainty of reputation opinions is determined by the communication model and can be calculated as: $1 - oc_{i,j}(t)$, where $o$ is the weight parameter. Thus, we can calculate the reputation value $R_{i,j}^{\text{loc}}(t)$, which represents the expected belief of edge node $i$ that edge node $j$ can provide real and relevant intelligence at time slot $t$, as follows:

$$R_{i,j}^{\text{loc}}(t) = RB_{i,j}^{\text{loc}}(t) + \iota RU_{i,j}^{\text{loc}}(t), \tag{8}$$

where $\iota \in [0,1]$ is the weight parameter that represents the impact of uncertainty on reputation [30].

Furthermore, to obtain a more accurate and reliable reputation, we consider different weights to formulate local opinions.

- *Interaction Effects:* To prevent negative intelligence interaction events, we assign negative interactions a higher weight than positive interactions in reputation calculations. The weights of positive and negative interactions can be expressed as $m^p$ and $m^n$, respectively. $m^p \ll m^n$ and $m^p + m^n = 1$.
- *Interaction Freshness:* The trustworthiness of edge nodes changes over time, and recent events with higher freshness have greater impact than past events. Hence, we define $z^{t-t'} (z \in (0,1))$ as the freshness decay coefficient to describe the freshness of interaction events that occur during the period of $t' \in [0,t]$.

Combining the above different weights, the local opinion of edge node $i$ for edge node $j$ can be recalculated as follows

$$\begin{cases} RB_{i,j}^{\text{loc}}(t) = \left(1 - RU_{i,j}^{\text{loc}}(t)\right)\frac{m^p\tilde{d}_{i,j}^*(t)}{\tilde{d}_{i,j}^*(t)+\tilde{q}_{i,j}^*(t)} \\ RD_{i,j}^{\text{loc}}(t) = \left(1 - RU_{i,j}^{\text{loc}}(t)\right)\frac{m^n\tilde{q}_{i,j}^*(t)}{\tilde{q}_{i,j}^*(t)+\tilde{q}_{i,j}^*(t)}, \\ RU_{i,j}^{\text{loc}}(t) = 1 - oc_{i,j}(t) \end{cases} \tag{9}$$

where $\tilde{d}_{i,j}^*(t) = \sum_{t'=0}^{t-1} z^{t-t'} d_{i,j}(t')$ and $\tilde{q}_{i,j}^*(t) = \sum_{t'=0}^{t-1} z^{t-t'} q_{i,j}(t')$.

### 3.4.2 Recommended Reputation

In addition to local opinions between edge nodes, recommended opinions from other edge nodes can also be considered. Some weights also need to be taken into account to build the recommended opinion model.

- *Interaction Frequency:* A higher interaction frequency signifies that edge node $i$ has more prior knowledge of edge node $j$, which leads to more accurate and reliable reputation calculations. Let $N_{i,j}(t) = \tilde{d}_{i,j}(t) + \tilde{q}_{i,j}(t)$ and $\bar{N}_i(t) = (1/\sum_{i' \in \mathcal{I}/\{i\}} \mathbf{1}(N_{i,j}(t) > 0)) \sum_{i' \in \mathcal{I}/\{i\}} N_{i,i'}(t)$, where $\mathbf{1}(\cdot)$ is the indicator function. The interaction frequency is the ratio of the number of interactions between edge node $i$ and edge node $j$ to the average number of interactions with other edge nodes, i,e., $\lambda_{i,j}(t) = N_{i,j}(t)/\bar{N}_i(t)$.
- *Node Correlation:* The node correlation will also affect the recommended opinion of edge nodes. When the data correlation $\zeta_{i,j}$ between edge node $i$ and edge node $j$ is higher, then the recommended opinion of edge node $j$ for edge node $i$ is more important.

Then, the recommended opinions can be integrated into a common opinion as $\Upsilon_{i,j}^{\text{rec}} := \{RB_{i,j}^{\text{rec}}, RD_{i,j}^{\text{rec}}, RU_{i,j}^{\text{rec}}\}$, where

$$\begin{cases} RB_{i,j}^{\text{rec}}(t) = \frac{1}{\sum\limits_{i' \in \mathcal{I}/\{i\}} \zeta_{i,i'}\lambda_{i,i'}(t)} \sum\limits_{i' \in \mathcal{I}/\{i\}} \zeta_{i,i'}\lambda_{i,i'}(t) RB_{i',j}(t) \\ RD_{i,j}^{\text{rec}}(t) = \frac{1}{\sum\limits_{i' \in \mathcal{I}/\{i\}} \zeta_{i,i'}\lambda_{i,i'}(t)} \sum\limits_{i' \in \mathcal{I}/\{i\}} \zeta_{i,i'}\lambda_{i,i'}(t) RD_{i',j}(t) \\ RU_{i,j}^{\text{rec}}(t) = \frac{1}{\sum\limits_{i' \in \mathcal{I}/\{i\}} \zeta_{i,i'}\lambda_{i,i'}(t)} \sum\limits_{i' \in \mathcal{I}/\{i\}} \zeta_{i,i'}\lambda_{i,i'}(t) RU_{i',j}(t) \end{cases} \tag{10}$$

#### 3.4.3 Composite Reputation

Integrating local reputation opinions and recommended reputation opinions, the composite reputation value of edge nodes can be calculated in the form of $\Upsilon_{i,j} := \{RB_{i,j}, RD_{i,j}, RU_{i,j}\}$, where

$$\begin{cases} RB_{i,j}(t) = \frac{RB^{\mathrm{loc}}_{i,j}(t)RU^{\mathrm{rec}}_{i,j}(t)+RB^{\mathrm{rec}}_{i,j}(t)RU^{\mathrm{loc}}_{i,j}(t)}{RU^{\mathrm{loc}}_{i,j}(t)+RU^{\mathrm{rec}}_{i,j}(t)-RU^{\mathrm{rec}}_{i,j}(t)RU^{\mathrm{loc}}_{i,j}(t)} \\ RD_{i,j}(t) = \frac{RD^{\mathrm{loc}}_{i,j}(t)RU^{\mathrm{rec}}_{i,j}(t)+RD^{\mathrm{rec}}_{i,j}(t)RU^{\mathrm{loc}}_{i,j}(t)}{RU^{\mathrm{loc}}_{i,j}(t)+RU^{\mathrm{rec}}_{i,j}(t)-RU^{\mathrm{rec}}_{i,j}(t)RU^{\mathrm{loc}}_{i,j}(t)} \\ RU_{i,j}(t) = \frac{RU^{\mathrm{loc}}_{i,j}(t)RU^{\mathrm{rec}}_{i,j}(t)}{RU^{\mathrm{loc}}_{i,j}(t)+RU^{\mathrm{rec}}_{i,j}(t)-RU^{\mathrm{rec}}_{i,j}(t)RU^{\mathrm{loc}}_{i,j}(t)} \end{cases} \tag{11}$$

Hence, the composite reputation value of edge node $i$ for edge node $j$ is $R_{i,j}(t) = RB_{i,j}(t) + \iota RU_{i,j}(t)$.

### 3.5 Intelligence Sharing Model

In our study, we use the model parameter vector $\boldsymbol{\omega}$ of a trained model as a type of intelligence that can be delivered and shared. Let $\varpi$ denote the number of required CPU cycles for edge nodes to process a data sample. The energy consumption of edge node $i$ for training its local model at time slot $t$ is

$$EC_i(t) = e_i \tau_i(t) \varpi \Lambda_i(t) (\Phi_i(t))^2, \tag{12}$$

where $e_i$ denotes the effective capacitance coefficient of edge node $i$'s computing chipset.

After completing the local training, edge nodes can decide whether to share their intelligence. The sharing of intelligence between edge nodes is paid, and the compensation for unit intelligence (intelligence per unit accuracy) $\kappa$ is related to the reputation and data relevance between edge nodes. According to the compensation, each edge node will decide which edge node it is willing to share intelligence for its benefit. We denote $b_{i,j}(t) \in \{0, 1\}$ as the indicator of intelligence sharing willingness, i.e., $b_{i,j}(t) = 1$ when edge node $i$ is willing to share its intelligence with edge node $j$. Let $TH_i$ represents the compensation threshold of edge node $i$. Therefore, we have $b_{i,j}(t) = 1$ when $\kappa\zeta_{i,j}R_{i,j}(t) \geq TH_i$; otherwise $b_{i,j}(t) = 0$. Then, each edge node can request intelligence sharing from the edge nodes that are willing to share the intelligence with it. Denote $r_{i,j}(t) \in \{0, 1\}$ as the indicator of intelligence requesting, where $r_{i,j}(t) = 1$ means that edge node $i$ asks edge node $j$ for intelligence sharing; otherwise, $r_{i,j}(t) = 0$. Specially, $r_{i,j}(t) = 1$, $\forall i = j$, $\forall t$. Moreover, we have $r_{j,i}(t) \leq b_{i,j}(t)$, $\forall t$. We consider the data size of a local model is $\varrho$ which is a constant with the same value for all edge nodes. Then, the energy consumption of edge node $i$ for routing its intelligence to other edge nodes can be calculated as

$$ET_i(t) = \sum_{j \in \mathcal{I}} \frac{p_{i,j}(t) \varrho r_{j,i}(t)}{c_{i,j}(t)}, \tag{13}$$

where $c_{i,j}(t)$ is the corresponding channel transmission rate between edge node $i$ and edge node $j$.

When edge node $i$ receives all the shared intelligence from other edge nodes, it first judges the positivity of the received intelligence. If the edge node $i$ judges that the intelligence received from edge node $j$ is useful, trusted, and reliable, it believes that this intelligence interaction with edge node $j$ to be a positive event, that is, $d_{i,j}(t) = 1$, otherwise $q_{i,j}(t) = 1$. Then, edge node $i$ aggregates positive intelligence models to obtain its aggregated model

$$\hat{\boldsymbol{\omega}}_i = \sum_{j \in \mathcal{I}} \frac{\varphi_j(t)}{\hat{\varphi}_i(t)} d_{i,j}(t) r_{i,j}(t) \boldsymbol{\omega}_j, \tag{14}$$

where $\hat{\varphi}_i(t) = \sum_{j \in \mathcal{I}} d_{i,j}(t) r_{i,j}(t) \varphi_j(t)$. $\boldsymbol{\omega}_j$ is the parameter vector of edge node $j$'s local model and $\hat{\boldsymbol{\omega}}_i$ is the parameter vector of edge node $i$'s aggregated model.

According to [34], we can derive the model performance of the aggregated model as

$$\epsilon_i(t) = 1 - \exp(-\varsigma^{\mathrm{ag}} \Phi_i(t) \tilde{\varphi}_i(t)), \tag{15}$$

where $\varsigma^{\mathrm{ag}}$ is a constant related to the computing capability of edge nodes, and $\tilde{\varphi}_i(t)$ is the average accuracy of models shared by other edge nodes to edge node $i$, which can be calculated as

$$\tilde{\varphi}_i(t) = \frac{\sum_{j \in \mathcal{I}} d_{i,j}(t) r_{i,j}(t) \varphi_j(t)}{\sum_{j \in \mathcal{I}} d_{i,j}(t) r_{i,j}(t)}. \tag{16}$$

## 4 Intelligence Sharing Optimization Problem

In this section, the problem statement is first presented. Then, the distributed intelligence sharing optimization problem is formulated as a multi-agent MDP.

### 4.1 Problem Statement

Based on the above system description, we are able to formulate our distributed intelligence sharing optimization problem. The main goal of this study is to develop a decision-maker for each edge node, who determines how to train AI models through intelligence sharing such that the obtained utility of training AI models is maximized. In our system, time is logically slotted. The system state comprises the volume of collected data, the quality level of collected data, the computing capability of edge nodes, the channel conditions of edge networks and the reputation opinions between edge nodes. At the beginning of each time slot, each edge node independently makes a training iteration selection decision to determine the training level of its local model (that is, how many training iterations are required to train the local model) based on its observation on the current system state. Given the training iteration selection decision, the edge nodes will consume energy resources and computing resources to train their local AI model. According to the trained intelligence, each edge node makes the intelligence requesting decision and spectrum resource allocation decision to share its intelligence with other edge nodes in the edge network. Then, an immediate utility is obtained, which is a trade-off between the performance of trained AI models, the revenue and cost for sharing intelligence with other edge nodes and requesting intelligence from other edge nodes, and the energy consumption for training and transmitting AI models. Afterwards, the network environment evolves to the next state.

In the following, we will formulate the distributed intelligence sharing optimization problem as a multi-agent MDP, which provides optimal intelligence sharing policies for edge nodes at every system state.

## 4.2 Problem Formulation

We leverage a multi-agent MDP framework to formulate our distributed intelligence sharing optimization problem [35], [36]. An multi-agent MDP formulation can be presented as a tuple $< \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{U} >$ , where $\mathcal{I}$ is the set of $I$ learning agents; $\mathcal{S}$ the set of possible states, $\mathcal{A}$ denotes the set of joint actions, $\mathcal{P}$ is a state transition probability, $\mathcal{U}$ is the utility function.

### 4.2.1 State Space

In each time slot, the edge nodes observe the Internet of intelligence-empowered edge computing environment and collect the following parameters:

- the volume of collected data samples at $t$th time slot $\Lambda_i(t), \forall i \in \mathcal{I}$;
- the quality level of collected data samples at $t$th time slot $\Upsilon_i(t), \forall i \in \mathcal{I}$;
- the available computing capability of edge nodes at $t$th time slot $\Phi_i(t), \forall i \in \mathcal{I}$;
- the channel conditions at $t$th time slot $\Omega_{i,j}(t), \forall i, j \in \mathcal{I}$;
- the reputation opinion conditions at $t$th time slot $R_{i,j}(t), \forall i, j \in \mathcal{I}$.

Therefore, the composite state $\boldsymbol{s}_t \in \mathcal{S}$ can be described as: $\boldsymbol{s}_t = (\boldsymbol{s}_{1,t}, \dots, \boldsymbol{s}_{i,t}, \dots, \boldsymbol{s}_{I,t})$. Here, $\boldsymbol{s}_{i,t}$ is the system state of edge node $i$ at time slot $t$, and can be denoted as

$$\begin{aligned} \boldsymbol{s}_{i,t} = [&\Lambda_i(t), \Upsilon_i(t), \Phi_i(t), \\ &\Omega_{i,1}(t), \dots, \Omega_{i,j}(t), \dots, \Omega_{i,I}(t), \\ &R_{i,1}(t), \dots, R_{i,j}(t), \dots, R_{i,I}(t)]. \end{aligned} \tag{17}$$

### 4.2.2 Action Space

After observing the system state $\boldsymbol{s}_t \in \mathcal{S}$, the edge nodes will take intelligence sharing actions. The global action space $\mathcal{A}$ of edge nodes is defined as $\mathcal{A} = \{\mathcal{A}_1, \dots, \mathcal{A}_i, \dots, \mathcal{A}_I\}$. $\mathcal{A}_i$ is the local action space for edge node $i$, which includes the following:

- the training iteration selection at $t$th time slot $\tau_i(t)$;
- the intelligence requesting strategy at $t$th time slot $r_{i,j}(t), \forall j \in \mathcal{I}$;
- the spectrum resource allocation strategy at $t$th time slot $\eta_{i,j}(t), \forall j \in \mathcal{I}$.

Thus, the system action $\boldsymbol{a}_{i,t} \in \mathcal{A}_i$ of edge node $i$ at time slot $t$ can be represented as

$$\begin{aligned} \boldsymbol{a}_{i,t} &= [\tau_i(t), r_{i,1}(t), \dots, r_{i,j}(t), \dots, r_{i,I}(t), \\ &\quad \eta_{i,1}(t), \dots, \eta_{i,j}(t), \dots, \eta_{i,I}(t)]. \end{aligned} \tag{18}$$

The action is constrained and needs to satisfy the following conditions. The first condition is that edge node cannot request intelligence from other edge nodes that are unwilling to share intelligence with it. The second condition ensure that edge node cannot allocate a number of spectrum resources beyond their available resources.

### 4.2.3 Utility Function

In the intelligence sharing optimization problem, we aim to optimize the intelligence sharing decision including training iteration selection, intelligence requesting and spectrum resource allocation to maximize edge nodes' training utility according to the data quantity state, the data quality level state, the communication channel state, the computing capability state and the reputation state of the considered edge network.

For each edge node $i$, once it takes action $\boldsymbol{a}_{i,t}$ based on the observed environment state $\boldsymbol{s}_t$, the environment will feedback an immediate utility. The utility of edge nodes consists of two parts, i.e., revenue and cost. The revenue of edge nodes includes the revenue of the aggregated AI model and the revenue obtained by sharing intelligence to other edge nodes. The cost of edge nodes includes the cost of obtaining intelligence from other edge nodes, the cost of routing the intelligence to other edge nodes, and the cost of training local AI models. Accordingly, the immediate utility of edge node $i$, $i \in \mathcal{I}$ at time slot $t$ can be calculated as

$$\begin{aligned} u_{i,t} = \rho\epsilon_i(t) &+ \sum_{j \in \mathcal{I}/\{i\}} \kappa\zeta_{i,j} R_{j,i}(t) r_{j,i}(t) \varphi_i(t) \\ &- \sum_{j \in \mathcal{I}/\{i\}} \kappa\zeta_{i,j} R_{i,j}(t) r_{i,j}(t) \varphi_j(t) - ET_i(t) - EC_i(t), \end{aligned} \tag{19}$$

where $\rho$ is the performance gain coefficient and $\kappa$ is the price coefficient of the unit accuracy of intelligence. Therefore, $\rho\epsilon_i(t)$ is the revenue of the aggregated AI model. $\sum_{j \in \mathcal{I}/\{i\}} \kappa\zeta_{i,j} R_{j,i}(t) r_{j,i}(t) \varphi_i(t)$ and $\sum_{j \in \mathcal{I}/\{i\}} \kappa\zeta_{i,j} R_{i,j}(t) r_{i,j}(t) \varphi_j(t)$ are the revenue obtained by sharing intelligence to other edge nodes and the cost of obtaining intelligence from other edge nodes, respectively. $ET_i(t)$ and $EC_i(t)$ are the energy cost of routing the intelligence to other edge nodes and the energy cost of training local AI models, respectively.

Given the above unified utility model, we can formulate the intelligence sharing optimization problem as follows:

$$\max_{\tau_i(t), r_{i,j}(t), \eta_{i,j}(t)} \limsup_{T \to \infty} \frac{1}{T} \mathbb{E}\left[\sum_{t=0}^{T-1} u_{i,t}\right] \tag{20a}$$

$$\text{s.t.} \quad \tau_i(t) \geq 0, \forall t \tag{20b}$$

$$r_{i,j}(t) = \{0, 1\}, \forall j \in \mathcal{I}, \forall t \tag{20c}$$

$$r_{i,j}(t) \leq b_{j,i}(t), \forall j \in \mathcal{I}, \forall t \tag{20d}$$

$$0 \leq \eta_{i,j}(t) \leq 1, \forall j \in \mathcal{I}, \forall t \tag{20e}$$

$$\sum_{j \in \mathcal{I}} \eta_{i,j}(t) \leq 1, \forall t \tag{20f}$$

where $\mathbb{E}[\cdot]$ is the expected utility of edge nodes at the long run. Constraint (20b), (20c) and (20e) denote the value of training iteration decision $\tau_i(t)$, intelligence requesting decision $r_{i,j}(t)$, and spectrum allocation decision $\eta_{i,j}(t)$, respectively. Constraint (20d) is that the edge node cannot request intelligence from other edge nodes that are unwilling to share intelligence with it. Constraint (20f) ensure that edge nodes cannot allocate a number of spectrum resources beyond their available resources.

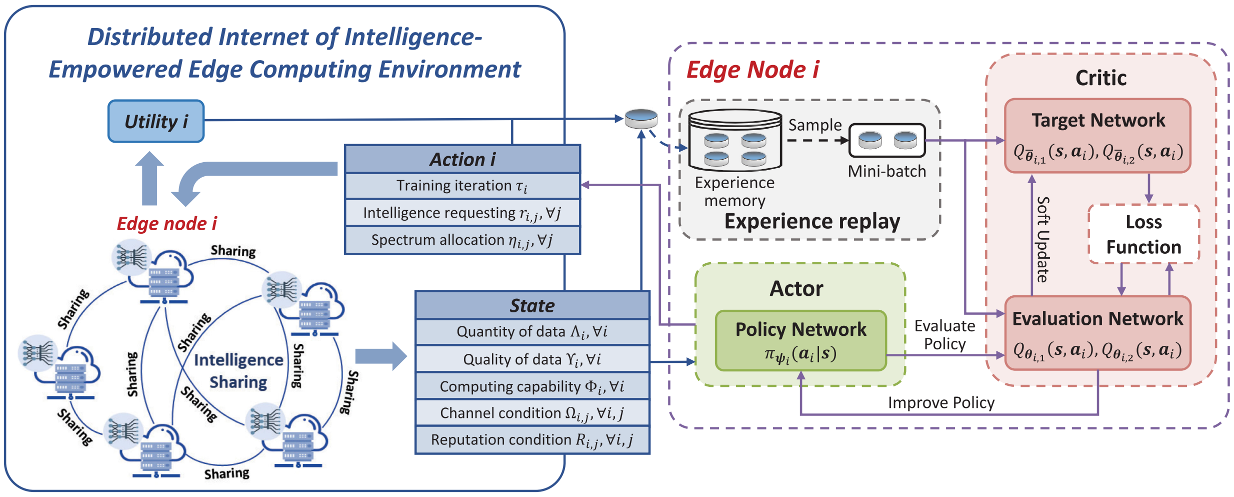

Distributed Internet of Intelligence-Empowered Edge Computing Environment
Utility i
Action i
Training iteration $\tau_i$
Intelligence requesting $r_{i,j}, \forall j$
Spectrum allocation $\eta_{i,j}, \forall j$
Edge node i
Sharing
Intelligence Sharing
State
Quantity of data $\Lambda_i, \forall i$
Quality of data $\Upsilon_i, \forall i$
Computing capability $\Phi_i, \forall i$
Channel condition $\Omega_{i,j}, \forall i, j$
Reputation condition $R_{i,j}, \forall i, j$
Edge Node i
Sample
Experience memory
Mini-batch
Experience replay
Critic
Target Network
$Q_{\bar{\theta}_{i,1}}(s, a_i), Q_{\bar{\theta}_{i,2}}(s, a_i)$
Soft Update
Loss Function
Evaluation Network
$Q_{\theta_{i,1}}(s, a_i), Q_{\theta_{i,2}}(s, a_i)$
Actor
Policy Network
$\pi_{\psi_i}(a_i|s)$
Evaluate Policy
Improve Policy

Fig. 2. The architecture of the proposed multi-agent SAC-based intelligence sharing algorithm.

## 5 Problem Solution With Collective Deep Reinforcement Algorithm

In this section, to solve the above multi-agent MDP formulation for distributed intelligence sharing, we design a novel CDRL algorithm consisting of local SAC learning at each edge node and collective learning between different edge nodes.

### 5.1 Local Learning With Multi-Agent Soft Actor-Critic

By extending the traditional reinforcement learning (RL) algorithm, we first employ a model-free SAC-based multi-agent DRL algorithm to implement local learning at each edge node, as illustrated in Fig. 2. In multi-agent SAC, there are three main components to improve performance [37]:

- *Actor-critic framework:* SAC is implemented based on the actor-critic framework, which consists of the actor part and the critic part. The actor is used to search for the best strategy to maximize the expected utility, while the critic estimates the state and state-action value. Supported by the actor-critic framework, SAC integrates the advantages of value-based and policy-based RL.
- *Maximum entropy:* Incorporating the entropy measurement of policies into the utility significantly improves the stochasticity of the SAC's policy and encourages the exploration of more possible optimal decisions. Therefore, compared with some other policy-based DRL algorithms, SAC has stronger robustness and generalization, making it easier to adjust in a stochastic intelligence sharing environment.
- *Off-policy formulation:* SAC adopts an off-policy formulation to train network parameters based on experience replay technique, thereby effectively using sampled experiences to achieve convergence.

In the following, we will introduce in detail how each edge node uses SAC for local learning. Specifically, we first introduce the definition of soft value functions of SAC and then present the learning process of the critic (policy evaluation) and the actor (policy improvement), respectively.

#### 5.1.1 *Soft Value Functions*

The objective of each edge node $i$ is to find a intelligence sharing policy $\pi_i(\boldsymbol{a}_i|\boldsymbol{s})$ to maximize the expected utility in the future while following this policy. In SAC-based DRL algorithm, an entropy $\mathcal{H}(\pi_i(\boldsymbol{a}_i|\boldsymbol{s})) = -\log \pi_i(\boldsymbol{a}_i|\boldsymbol{s})$ is added to the utility to ensure that edge node $i$ can explore continually [38]. The objective with expected entropy is termed as entropy objective and can be written as

$$\pi_i^* = \arg\max_{\pi_i} \mathop{\mathbb{E}}_{\boldsymbol{s}_t, \boldsymbol{a}_{i,t} \sim \pi_i} \left[ \sum_{t=0}^{\infty} \gamma^t \big( u_{i,t} + \alpha_i \mathcal{H}(\pi_i(\cdot|\boldsymbol{s}_t)) \big) \right], \tag{21}$$

where $\mathcal{H}(\cdot)$ calculates the entropy of policy $\pi_i$ as $\mathcal{H}(\pi_i(\cdot|\boldsymbol{s})) = \mathbb{E}_{\boldsymbol{a}_i}[-\log \pi_i(\boldsymbol{a}_i|\boldsymbol{s})]$; $\gamma$ is the discount factor for long-term utility calculation; $\alpha_i$ is the temperature parameter to control the magnitude of entropy regularization. Actually, it is a learnable parameter to balance the trade-off between exploration and exploitation.

To evaluate policy $\pi_i$ and improve it in the training, we first need to estimate the soft state-value function (i.e., the goodness of state $\boldsymbol{s}$) and the soft Q-value function (i.e., the long-term soft reward of taking action $\boldsymbol{a}_i$ at state $\boldsymbol{s}$). The soft state-value function takes into account the entropy-augmented accumulated return can be defined as follows:

$$V_i^{\pi}(\boldsymbol{s}) = \mathop{\mathbb{E}}_{\boldsymbol{s}_t, \boldsymbol{a}_{i,t} \sim \pi_i} \left[ \sum_{t=0}^{\infty} \gamma^t \big( u_{i,t} + \alpha_i H(\pi_i(\cdot \mid \boldsymbol{s}_t)) \big) \mid \boldsymbol{s}_0 = \boldsymbol{s} \right]. \tag{22}$$

Correspondingly, the soft Q-function can be given as

$$Q_i^{\pi}(\boldsymbol{s}, \boldsymbol{a}_i) = \mathop{\mathbb{E}}_{\boldsymbol{s}_t, \boldsymbol{a}_{i,t} \sim \pi_i} \left[ \sum_{t=0}^{\infty} \gamma^t u_{i,t} + \alpha_i \sum_{t=1}^{\infty} \gamma^t H(\pi_i(\cdot \mid \boldsymbol{s}_t)) \mid \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_{i,0} = \boldsymbol{a}_i \right]. \tag{23}$$

SAC uses the given policy $\pi_i$ to iteratively calculate the soft value functions. The relationship between $V_i^{\pi}(\boldsymbol{s})$ and $Q_i^{\pi}(\boldsymbol{s}, \boldsymbol{a}_i)$ can be shown as follows according to the Bellman equation

$$Q_i^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_{i,t}) = u_{i,t} + \mathop{\mathbb{E}}_{\boldsymbol{s}_{t+1}\sim\pi_i}\left[V_i^{\pi}(\boldsymbol{s}_{t+1})\right], \tag{24}$$

$$V_i^{\pi}(\boldsymbol{s}_t) = \mathop{\mathbb{E}}_{\boldsymbol{s}_t, \boldsymbol{a}_{i,t}\sim\pi_i}\left[Q_i^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_{i,t}) + \alpha\mathcal{H}(\pi_i(\cdot|\boldsymbol{s}_t))\right]. \tag{25}$$

Due to the high dimensionality of the state space and action space of the intelligence sharing problem, DNN is used in the algorithm to express the soft action-value function and intelligence sharing policy [39]. Then, by using a fully connected DNN with parameter $\boldsymbol{\theta}_i$ and $\boldsymbol{\psi}_i$, the soft action-value function $Q_i^{\pi}(\boldsymbol{s}, \boldsymbol{a}_i)$ and the policy $\pi_i(\boldsymbol{a}_i|\boldsymbol{s})$ can be parameterized as $Q_{\boldsymbol{\theta}_i}(\boldsymbol{s}, \boldsymbol{a}_i)$ and $\pi_{\boldsymbol{\psi}_i}(\boldsymbol{a}_i|\boldsymbol{s})$, respectively.

### 5.1.2 Policy Evaluation

In the training process, experience replay technique is adopted to disrupt the temporal correlations of samples. In each time slot, the transition of the network environment state, the actions performed and the immediate utility form a tuple $\{\boldsymbol{s}_t, \boldsymbol{a}_{i,t}, u_{i,t}, \boldsymbol{s}_{t+1}\}$, which is then stored in a fixed-size experience replay buffer $\mathcal{M}_r$. The actor and the critic are updated by randomly sampling a mini-batch $\mathcal{M}_b$ of tuples $\{\boldsymbol{s}_t^m, \boldsymbol{a}_{i,t}^m, u_{i,t}^m, \boldsymbol{s}_{t+1}^m\}_{i\in\mathcal{I}}$ from the buffer.

Two separate critic neural networks with parameters $\boldsymbol{\theta}_{i,1}$ and $\boldsymbol{\theta}_{i,2}$ are designed to avoid the overestimation of state values, and we take the smaller one of $Q_{\boldsymbol{\theta}_{i,1}}$ and $Q_{\boldsymbol{\theta}_{i,2}}$ as the actual Q-value. The parameters $\boldsymbol{\theta}_{i,k}$, $\forall k \in \{1, 2\}$ of the evaluation critic networks are independently updated by minimizing the loss function $L(\boldsymbol{\theta}_{i,k})$. To mitigate positive bias in policy improvement, a target soft action-value function with parameters $\bar{\boldsymbol{\theta}}_{i,k}$ is defined, which is defined as follows:

$$\ell_{i,t}^m = u_{i,t}^m + \gamma\Big(\min_{k=1,2} Q_{\bar{\boldsymbol{\theta}}_{i,k}}\left(\boldsymbol{s}_{t+1}^m, \boldsymbol{a}_{i,t+1}^m\right) - \alpha_i \log\left(\pi_{\boldsymbol{\psi}_i}\left(\boldsymbol{a}_{i,t+1}^m|\boldsymbol{s}_{t+1}^m\right)\right)\Big). \tag{26}$$

Then, the loss function of the critic can be given by

$$\mathcal{L}(\boldsymbol{\theta}_{i,k}) = \frac{1}{2\mathcal{M}_b}\sum_{m=1}^{\mathcal{M}_b}\left(\min_{k=1,2} Q_{\boldsymbol{\theta}_{i,k}}\left(\boldsymbol{s}_t^m, \boldsymbol{a}_{i,t}^m\right) - \ell_{i,t}^m\right)^2. \tag{27}$$

Thus, the stochastic gradient can be obtained for updating the critic networks

$$\nabla_{\boldsymbol{\theta}_{i,k}}\mathcal{L}(\boldsymbol{\theta}_{i,k}) = \frac{1}{\mathcal{M}_b}\sum_{k=1}^{\mathcal{M}_b}\nabla_{\boldsymbol{\theta}_{i,k}} Q_{\boldsymbol{\theta}_{i,k}}\left(\boldsymbol{s}_t^m, \boldsymbol{a}_{i,t}^m\right) \cdot \Big(Q_{\boldsymbol{\theta}_{i,k}}\left(\boldsymbol{s}_t^m, \boldsymbol{a}_{i,t}^m\right) - \Big(u_{i,t}^m + \gamma\Big(Q_{\bar{\boldsymbol{\theta}}_{i,k}}\left(\boldsymbol{s}_{t+1}^m, \boldsymbol{a}_{i,t+1}^m\right) - \alpha_i \log\left(\pi_{\boldsymbol{\psi}_i}\left(\boldsymbol{a}_{i,t+1}^m|\boldsymbol{s}_{t+1}^m\right)\right)\Big)\Big)\Big). \tag{28}$$

Moreover, in order to stabilize the learning process, the target critic network parameters $\bar{\boldsymbol{\theta}}_{i,k}$ are updated from the evaluation critic network parameters $\boldsymbol{\theta}_{i,k}$ through a soft-updating method, i.e.,

$$\bar{\boldsymbol{\theta}}_{i,k} = \vartheta\boldsymbol{\theta}_{i,k} + (1-\vartheta)\bar{\boldsymbol{\theta}}_{i,k}, \forall k = 1, 2, \tag{29}$$

where $\vartheta \in (0, 1)$ is the update factor.

### 5.1.3 Policy Improvement

If the current intelligence sharing policy is optimal, all intelligence models in a period will be trained at a lower cost and the total utility will be the largest. Conversely, if the current policy is poor, the training accuracy of some intelligence models may be low or require a high cost to complete, resulting in a small utility value. Therefore, policy improvement is of great significance. In the algorithm, we improve the policy parameters by minimizing the expected KL-divergence

$$J(\boldsymbol{\psi}_i) = \mathop{\mathbb{E}}_{\boldsymbol{s}_t\sim\mathcal{M}_b}\left[\mathrm{D_{KL}}\left(\pi_{\boldsymbol{\psi}_i}(\cdot|\boldsymbol{s}_t)\middle\|\frac{\exp\left(\min_{k=1,2} Q_{\boldsymbol{\theta}_{i,k}}(\boldsymbol{s}_t, \cdot)\right)}{Z_{\boldsymbol{\theta}_{i,k}}(\boldsymbol{s}_t)}\right)\right], \tag{30}$$

where the KL divergence $\mathrm{D_{KL}}(p\|q)$ measures the difference of distributions $p$ and $q$. $Z_{\boldsymbol{\theta}_{i,k}}(\boldsymbol{s}_t)$ is an intractable partition function that does not contribute to the new policy's gradient. Therefore, by multiplying a temperature parameter $\alpha_i$ and discarding the term $\mathbb{E}_{\boldsymbol{a}_t\sim\pi_{\boldsymbol{\psi}}}[\alpha_i \log Z_{\boldsymbol{\theta}_{i,k}}(\boldsymbol{s}_t)]$, the above KL-divergence can be further converted as follows:

$$J(\boldsymbol{\psi}_i) = -\mathop{\mathbb{E}}_{\boldsymbol{s}_{t+1}\sim\mathcal{M}_b}\left[\mathop{\mathbb{E}}_{\boldsymbol{a}_{t+1}\sim\pi_{\boldsymbol{\psi}_i}}\left[\min_{k=1,2} Q_{\bar{\boldsymbol{\theta}}_{i,k}}\left(\boldsymbol{s}_{t+1}^m, \boldsymbol{a}_{i,t+1}^m\right) - \alpha_i \log\left(\pi_{\boldsymbol{\psi}_i}\left(\boldsymbol{a}_{i,t+1}^m|\boldsymbol{s}_{t+1}^m\right)\right)\right]\right]. \tag{31}$$

Here, the policy is reparameterized via a neural network transformation denoted by $\delta_{\boldsymbol{\psi}_i}(\varepsilon_t; \boldsymbol{s}_t)$, where $\varepsilon_t$ is an input noise sampled from a Gaussian distribution. Accordingly, Eq. (26) can be rewritten as

$$J(\boldsymbol{\psi}_i) = -\mathop{\mathbb{E}}_{\boldsymbol{s}_{t+1}\sim\mathcal{M}_b, \varepsilon_t\sim\pi_{\boldsymbol{\psi}_i}}\left[\min_{k=1,2} Q_{\bar{\theta}_{i,k}}\left(s_{t+1}^m, \delta_{\boldsymbol{\psi}_i}\left(\varepsilon_t; s_t^m\right)\right) - \alpha_i \log\left(\pi_{\boldsymbol{\psi}_i}\left(\delta_{\boldsymbol{\psi}_i}\left(\varepsilon_t; s_t^m\right) \mid s_{t+1}^m\right)\right)\right]. \tag{32}$$

Thus, the gradient of the policy can be approximated with

$$\nabla_{\psi_i} J(\boldsymbol{\psi}_i) = \nabla_{\boldsymbol{\psi}_i}\alpha_i \log\left(\pi_{\boldsymbol{\psi}_i}\left(\boldsymbol{a}_{i,t}^m|\boldsymbol{s}_t^m\right)\right) + \left(\nabla_{\boldsymbol{\psi}_i}\alpha_i \log\left(\pi_{\boldsymbol{\psi}_i}\left(\boldsymbol{a}_{i,t}^m|\boldsymbol{s}_t^m\right)\right) - \nabla_{\boldsymbol{\psi}_i} Q\left(\boldsymbol{s}_t^m, \boldsymbol{a}_{i,t}^m\right)\right)\nabla_{\boldsymbol{\psi}_i}\delta_{\boldsymbol{\psi}_i}(\varepsilon_t; \boldsymbol{s}_t^m). \tag{33}$$

The soft policy evaluation and the soft policy improvement alternate in the learning process until the iteration converges to an optimal policy with the largest entropy.

### 5.1.4 Multi-Agent Soft Actor-Critic Algorithm

The details of the proposed SAC-based local learning algorithm for edge nodes are shown in Algorithm 1. In the following, we illustrate the main steps of the algorithm:

- *Step 1:* At the beginning of the training stage, the parameters of the soft Q-value functions $Q_{\boldsymbol{\theta}_{i,k}}(\boldsymbol{s}_t, \boldsymbol{a}_{i,t})$, $\forall k \in \{1, 2\}$ are initialized with weights $\boldsymbol{\theta}_{i,k}$. The parameters of target soft Q-value $\bar{\boldsymbol{\theta}}_{i,k}$ are given with the weights $\boldsymbol{\theta}_{i,k}$. The parameters of the policy

$\pi_{\psi_i}(\boldsymbol{a}_i|\boldsymbol{s})$ are initialized with random weights and an experience replay memory $\mathcal{M}_r$ is instantiated.

- *Step 2:* In each episode, the edge node first observes the initial state $\boldsymbol{s}_0$ by observing the Internet of intelligence-empowered edge computing environment.
- *Step 3:* In each time slot, each edge node observes the network state information $\boldsymbol{s}_t$. Then the actors of edge nodes independently generate action $\boldsymbol{a}_{i,t}$ according to state $\boldsymbol{s}_t$ and policy $\pi_{\psi_i}(\boldsymbol{a}_{i,t}|\boldsymbol{s}_t)$. Given the selected action, each edge node obtain the feedback utility $u_{i,t}$, and network environments evolve to the next state $\boldsymbol{s}_{t+1}$.
- *Step 4:* The experience of the edge nodes is denoted by a tuple $\{\boldsymbol{s}_t, \boldsymbol{a}_{i,t}, u_{i,t}, \boldsymbol{s}_{t+1}\}_{i\in\mathcal{I}}$ of the performed action, state transition, and feedback utility, and then is stored into the experience replay memory $\mathcal{M}_r$.
- *Step 5:* Each edge node $i$ randomly selects a minibatch of experiences $\mathcal{M}_b$ from $\mathcal{M}_r$. Then, the parameters $\boldsymbol{\theta}_{i,k}$ of soft Q-value functions and the parameters $\boldsymbol{\psi}_i$ of policy are are updated by minimizing the loss function $\mathcal{L}(\boldsymbol{\theta}_{i,k})$ and $J(\boldsymbol{\psi}_i)$ defined in (27) and (32), respectively. Besides, the temperature parameters can be updated automatically by calculating to the gradient of the following objective

$$J(\alpha_i) = \mathop{\mathbb{E}}_{\boldsymbol{a}_t\sim\pi_{\boldsymbol{\theta}_i}}\left[-\alpha_i \log\left(\pi_{\boldsymbol{\theta}_i}\left(\boldsymbol{a}_t^m|\boldsymbol{s}_t^m\right)\right) - \alpha_i\bar{H}\right], \tag{34}$$

  where $\boldsymbol{a}_t^m = \{\boldsymbol{a}_{i,t}^m\}$ is the action set of all edge nodes, and $\bar{H}$ is the value of target entropy.
- *Step 6:* Each edge node $i$ updates the parameters of target critic network $\bar{\boldsymbol{\theta}}_{i,k}$ from the evaluation critic network parameters $\boldsymbol{\theta}_{i,k}$ according to Eq. (29).

## 5.2 Collective Learning Between Different Edge Nodes

Through the above multi-agent SAC-based local learning algorithm, each edge node can find an optimal intelligence sharing policy through continuous iteration to maximize its utility in a previously unknown environment via its own experience. The optimization objective consists of two parts, i.e., exploitation and exploration. The purpose of exploitation is to use known environmental information to maximize utilities, while exploration encourages actions that have not been tried. In the above multi-agent SAC-based algorithm, temperature parameter $\alpha_i$ is an exploration-exploitation trade-off coefficient. A larger $\alpha_i$ will encourage edge node $i$ to conduct more random exploration in the learning process. Denote $\mathcal{P}_{\psi_{i,t}}$ as the model we are learning through Algorithm 1. According to [40], the optimization of the SAC-based algorithm can then be rewritten as follows:

$$\max_{\pi_i} \underbrace{u_{i,t}}_{\text{Exploitation}} + \underbrace{\alpha_i \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_{i,t}\sim\pi_i}\left[\mathrm{D_{KL}}\left(\mathcal{P}_i||\mathcal{P}_{\psi_{i,t}}\right)[\boldsymbol{s}_t, \boldsymbol{a}_{i,t}]\right]}_{\text{Exploration}}, \tag{35}$$

where $\mathcal{P}_i$ is the practical model. The exploration incentive is an average KL-divergence of $\mathcal{P}_i$ from $\mathcal{P}_{\psi_{i,t}}$, aiming to capture the edge node's surprise about its intelligence sharing experience.

**Algorithm 1.** Multi-Agent SAC-Based Local Learning Algorithm

```
Input: pre-train episodes, pre-train time slots, discount
       factor, replay memory size, minibatch size.
1  Initialize the evaluation critic with parameters $\boldsymbol{\theta}_{i,k}$
   ($k = 1, 2$), the target critic with $\bar{\boldsymbol{\theta}}_{i,k} = \boldsymbol{\theta}_{i,k}$, the actor with
   parameters $\boldsymbol{\psi}_i$, the temperature parameter $\alpha_i$, and the replay
   memory $\mathcal{M}_r$ for every edge node $i \in \mathcal{I}$;
2  for each episode do
3    Initialize the network state $\boldsymbol{s}_0 = \{\boldsymbol{s}_{i,0}\}_{\forall i\in\mathcal{I}}$;
4    for each time slot $t$ do
5      for each edge node $i$ do
6        ********* Experience sampling *********
7        Observe the system and retrieve
         $\boldsymbol{s}_t = [\Lambda_i(t), \Upsilon_i(t), \Phi_i(t), \Omega_{i,j}(t), R_{i,j}(t)], \forall i, j \in \mathcal{I}$;
8        Obtain intelligence sharing decision $\boldsymbol{a}_{i,t} =$
         $[\tau_i(t), r_{i,j}(t), \eta_{i,j}(t)], \forall j \in \mathcal{I}$ with policy $\pi_{\psi_i}(\boldsymbol{a}_{i,t}|\boldsymbol{s}_t)$;
9        Train the local model according to $\tau_i(t)$, route the
         shared intelligence to other edge nodes according to
         $r_{i,j}(t)$ and allocate the spectrum resources according
         to $\eta_{i,j}(t)$;
10       Aggregate positive intelligence models according to
         Eq. (14);
11       Observe immediate utility $u_{i,t}$ and obtain the next
         state $\boldsymbol{s}_{t+1}$ through message passing;
12       Store the four tuple $\{\boldsymbol{s}_t, \boldsymbol{a}_{i,t}, u_{i,t}, \boldsymbol{s}_{t+1}\}$ and replace the
         oldest experiences in $\mathcal{M}_r$;
13       ********* Parameter updating *********
14       Randomly sample a minibatch of $\mathcal{M}_b$ experiences
         $\{\boldsymbol{s}_t, \boldsymbol{a}_{i,t}, u_{i,t}, \boldsymbol{s}_{t+1}\}$ from $\mathcal{M}_r$;
15       Compute the target Q-value $\ell_{i,t}^k$ by Eq. (26);
16       Update soft Q-value function parameters $\boldsymbol{\theta}_{i,k}$ by
         minimizing the loss function in Eq. (27);
17       Update the intelligence sharing policy parameters $\boldsymbol{\psi}_i$
         via the gradient in Eq. (33);
18       Update temperature parameter $\alpha_i$ by calculating the
         gradient defined in Eq. (34);
19       Update target networks with Eq. (29);
Output: intelligence sharing policy $\boldsymbol{a} = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_I\}$.
```

Exploitation and exploration are conducted in the local environment, so the edge node needs massive pre-defined datasets with the local intelligence sharing environment. To achieve a higher-quality exploration with limited datasets, combining the idea of "collective learning" with the proposed multi-agent SAC-based DRL algorithm, we propose to use a novel concept of learning that works between different edge nodes, named CDRL. Indeed, DRL based on "collective learning" has been discussed in existing work [41]. However, "collective learning" in such work mainly focuses on collecting and sharing experience tuples involved in training. As a result, distributed agents still need to expend significant computing and energy resources to learn from shared experience tuples. In contrast, by introducing the concept of *extension*, the CDRL proposed in this paper is based on the idea of "collective intelligence". *Extension* encourages edge nodes to share learned intelligence with others when making decisions and adjusts the utility function of the SAC-based DRL algorithm by executing a KL-divergence of the practical model from the ideal model. Denote $\beta_i$ as the coefficient of extension, and then the optimization can be given as follows [42]

$$\max_{\pi_i} \underbrace{u_{i,t}}_{\text{Exploitation}} + \underbrace{\alpha_i \mathbb{E}_{\boldsymbol{s}_t, \boldsymbol{a}_{i,t} \sim \pi_i} \left[ \mathrm{D}_{\mathrm{KL}} \left( \mathcal{P}_i || \mathcal{P}_{\boldsymbol{\psi}_{i,t}} \right) [\boldsymbol{s}_t, \boldsymbol{a}_{i,t}] \right]}_{\text{Exploration}}$$

$$- \underbrace{\beta_i \mathbb{E}_{\boldsymbol{s}_t, \boldsymbol{a}_{i,t} \sim \pi_i} \left[ \mathrm{D}_{\mathrm{KL}} \left( \mathcal{P}_i || \tilde{\mathcal{P}}_{-i} \right) [\boldsymbol{s}_t, \boldsymbol{a}_{i,t}] \right]}_{\text{Extension}} \quad (36)$$

where $\tilde{\mathcal{P}}_{-i}$ is the dynamic model of other edge nodes except edge node $i$. With the exploration and extension incentives, the utility function can be approximated as

$$\begin{aligned} u'_{i,t} = u_{i,t} &+ \alpha_i \Big[ \log \mathcal{P}_{\boldsymbol{\psi}_{i,t}} \big( \boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_{i,t} \big) \\ &- \log \mathcal{P}_{\boldsymbol{\psi}_{i,t-l}} \big( \boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_{i,t} \big) ] - \beta_i \big[ \log \mathcal{P}_{\boldsymbol{\psi}_{i,t}} \big( \boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_{i,t} \big) \\ &- \log \tilde{\mathcal{P}}_{\boldsymbol{\psi}_{-i,t-l+1}} \big( \boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_{i,t} \big) ]. \end{aligned} \quad (37)$$

Here, $u_{i,t}$ is the original utility and $u'_{i,t}$ is the collective utility. By replacing the utility function $u_{i,t}$ in Algorithm 1 with the collective utility $u'_{i,t}$, we can obtain the CDRL algorithm for the intelligence sharing optimization problem.

The resource and energy consumption of Internet of intelligence empowered edge nodes are the indicators that our work expects to optimize. Due to the high spatial and temporal complexity of neural networks, the training process of the proposed CDRL algorithm may require a large amount of computing resources and energy consumption. Fortunately, the proposed SAC-based algorithm is an offline algorithm [38], [43]. Accordingly, in practice, the proposed SAC-based CDRL algorithm can be pre-trained offline at edge nodes. Then, optimal intelligence sharing decisions can be made in an online manner with a small amount of resources and energy directly using well-trained algorithms.

## 6 Simulation Results and Discussions

In this section, extensive simulations have been conducted to evaluate the performance of our proposed distributed intelligence sharing scheme in the Internet of intelligence-empowered edge computing scenarios.

### 6.1 Simulation Setup

The parameters for the experiment are set with reference to works [44], [45], [46], [47], [48]. We consider an Internet of intelligence-empowered edge network consisting of 10 edge nodes. The average computing capability of each edge node is set as 5 Gcycles/s. For the image data samples, we assume their size are common, i.e., 10 kB per image, which is on a par with the average image size of the well-known Cifar-10 data set [44]. The number of CPU cycles required to process each data sample follows the uniform distribution from 0.15 Gcycles to 0.45 Gcycles. The average parameter size of the trained intelligence model follows the uniform distribution from 200 KB to 300 KB. The number of data samples collected by the edge nodes follows the uniform distribution in the range of $[100, 300]$, and the average quality of collected data samples follows the uniform distribution in the range of $[0.65, 0.95]$. The performance gain coefficient and the price coefficient of the unit accuracy of intelligence are set to 10 and 5, respectively. Moreover, we assume that the available spectrum bandwidth between edge nodes is 20 MHz. The average transmitting power of edge nodes and the corresponding noise power are set to 100 mwatts and $-100$ dBm, respectively.

For the proposed CDRL-based intelligence sharing algorithm, the model is trained in iterations. The numerical results of the proposed scheme are collected after 300 learning episodes, and each episode consists of 100 time slots. The utility is accumulated in an episode. The development environment of CDRL is built on the top of Pytorch framework [49]. The learning rate (LR) is set as 0.0005 and the discount rate is set as 0.85. In order to calculate the loss function, a minibatch of tuples is randomly sampled from a replay memory of 10000 prior experiences, and the size pf minibatch is set as 256. All the numerical results presented in our work are obtained through multiple experiments to ensure estimation accuracy.

To evaluate the superiority of our proposed scheme, the following benchmark schemes are adopted for comparison:

- *SAC-based scheme*: at every decision period, each edge node makes the intelligence sharing decision based on the SAC method.
- *DDPG-based scheme*: at every decision period, the deep deterministic policy gradient (DDPG) method [50] is adopted by each edge node to select intelligence sharing actions.
- *Full sharing scheme*: at every decision period, each edge node always shares its intelligence with all other edge nodes.
- *Random sharing scheme*: each edge node randomly selects edge nodes to share its intelligence at every decision period.
- *Non-sharing scheme*: each edge node independently trains its intelligence model without the assistance of other edge nodes at every decision period.

The following metrics are used to evaluate the performance of our proposed scheme and benchmark schemes:

- *The total utility*: is the sum of edge nodes' utilities obtained by intelligence sharing during the simulation time.
- *The total accuracy gain*: refers to the weighted accuracy of the intelligence models trained by edge nodes.
- *The total sharing revenue*: refers to the total obtained revenue of edge nodes through sharing intelligence with other edge nodes.
- *The total energy consumption*: is the total energy consumed by edge nodes to train intelligence models and route intelligence parameters.

### 6.2 Simulation Results

Based on the above parameter settings, we conduct simulations to evaluate the proposed CDRL-based intelligence sharing scheme. We first test the feasibility of the proposed scheme in terms of the total utility of all edge nodes in the network. In Fig. 3, we study the convergence performance of the proposed scheme compared with the benchmark DRL schemes such as the SAC-based scheme and the DDPG-based scheme. As shown in the figure, the total utility of all three schemes gradually increases and converges to an approximately optimal value. Moreover, with the number of episodes increasing, the speed of convergence tends to

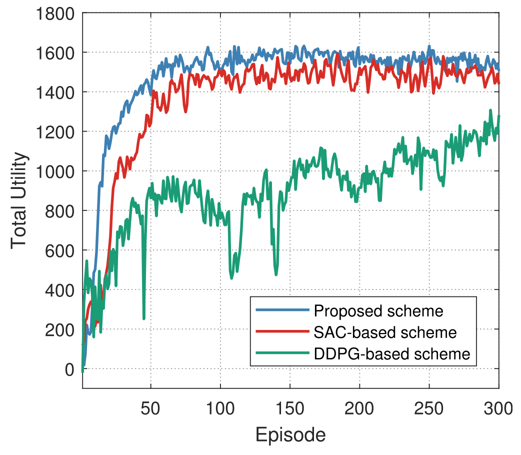


Fig. 3. Convergence performance of different intelligence sharing schemes.

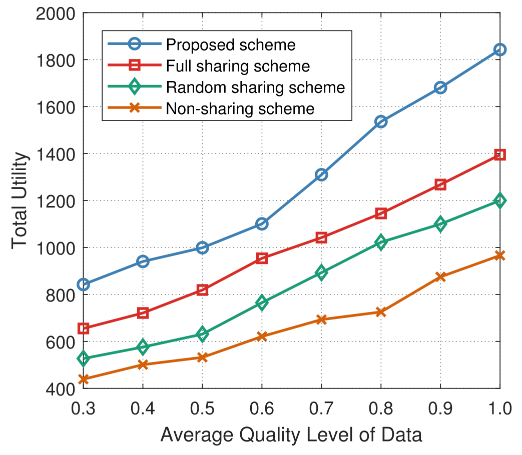


Fig. 5. Performance of different intelligence sharing schemes for the given average quality level of data.

increase. Though the DDPG is designed to optimize the expected long-term return, the total utility of the DDPG-based scheme is slightly lower than the total utility achieved by the other two DRL-based schemes. This significant improvement of the proposed scheme and the SAC-based scheme is attributed to the adoption of maximum entropy regularized stochastic policy rather than deterministic policy, because the latter is easy to overfit the value function and lead to extreme vulnerability. Moreover, our proposed scheme can obviously obtain a faster convergence speed and better utility than the SAC-based scheme. The convergence of the proposed CDRL-based scheme is achieved after around 80 episodes. This is because the proposed scheme enables the edge node to gain experience by consulting their neighboring edge nodes, thereby avoiding convergence to a local optimum.

In Fig. 4, we verify the impact of learning rates on algorithm performance. As shown in the figure, with the improvement of the learning rate, the proposed scheme has a faster convergence speed. A low learning rate leads to slow convergence speed, making the agent unable to learn from experience. However, fast convergence may make the edge node fall into the local optimum rather than the global optimum. Therefore, the learning rate should not be set too high or too low. In our work, after repeated experiments,

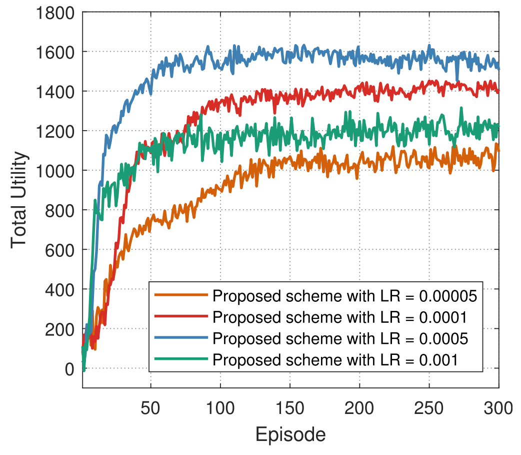


Fig. 4. Convergence performance of proposed scheme under different learning rate.

we choose 0.0005 as the fixed learning rate in the follow-up simulations because it has moderate convergence speed and good learning stability.

Then, we compare the proposed scheme with different intelligence sharing schemes to verify its effectiveness. In Fig. 5, we test the performance of the proposed scheme, the full sharing scheme, the random sharing scheme and the non-sharing scheme in terms of different quality levels of collected data. As shown in the figure, the total utility of all edge nodes always increases with the increase of the quality level of data for the four schemes as expected. The reason is as follows. Increasing the quality level of data has a direct impact on the accuracy of the trained model. The higher the quality level of the collected data, the more accurate the trained model. In addition, due to the increase in the quality level of data, edge nodes can greatly improve the accuracy of the model without training a large number of iterations, thereby reducing the energy consumption required to train models to a certain extent. In this way, the total utility is improved. We can also observe that compared with the benchmark intelligence sharing scheme, the proposed scheme can obtain a higher total utility for edge nodes through more flexible sharing decision-making. Specifically, when the quality level of data is 0.7, the utility obtained by the proposed scheme is about 1.26 times, 1.47 times, and 1.89 times that of the full sharing scheme, the random sharing scheme and the non-sharing scheme, respectively. The full sharing scheme can obtain a second-most performance, which is approximately 1.17 and 1.50 times that of the random sharing scheme and the non-sharing scheme when the quality level of data is 0.7, respectively. The non-sharing scheme enables edge nodes to only rely on their own data and computing resources for model training, so that it always achieve the worst performance.

In Fig. 6, we test the performance of the proposed scheme, the full sharing scheme, the random sharing scheme and the non-sharing scheme under different performance gain coefficients. The performance gain coefficient varies from 6 to 18 with a step size of 2. It can be seen that as the performance gain coefficients increase, the total utility of all intelligence sharing schemes will increase. This is because the increased performance gain coefficient will increase the total utility of edge nodes by improving the

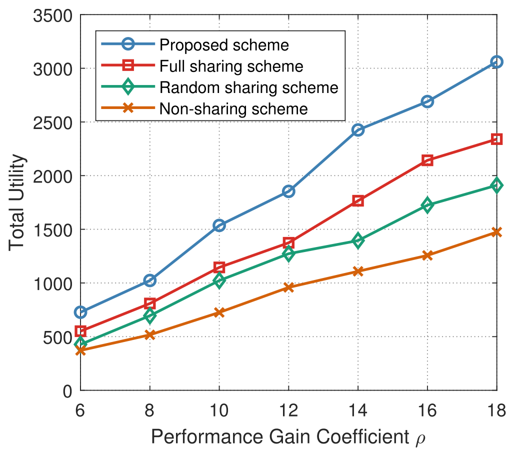
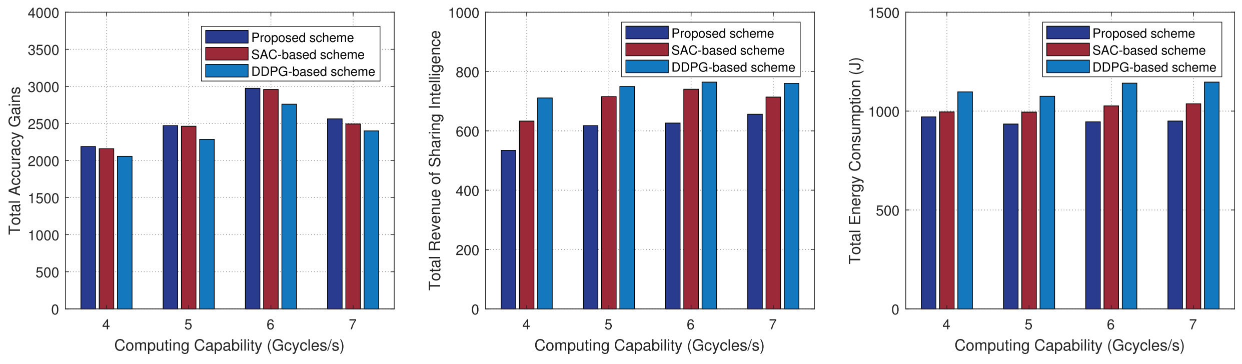


Fig. 6. Performance of different intelligence sharing schemes for the given performance gain coefficient.

obtained revenue brought by the trained models with the same accuracy. The proposed scheme still shows superior performance. When the performance gain coefficient is 12, the proposed scheme improves the performance of 1.35 times, 1.46 times and 1.93 times respectively compared with the full sharing scheme, the random sharing scheme and the non-sharing scheme.

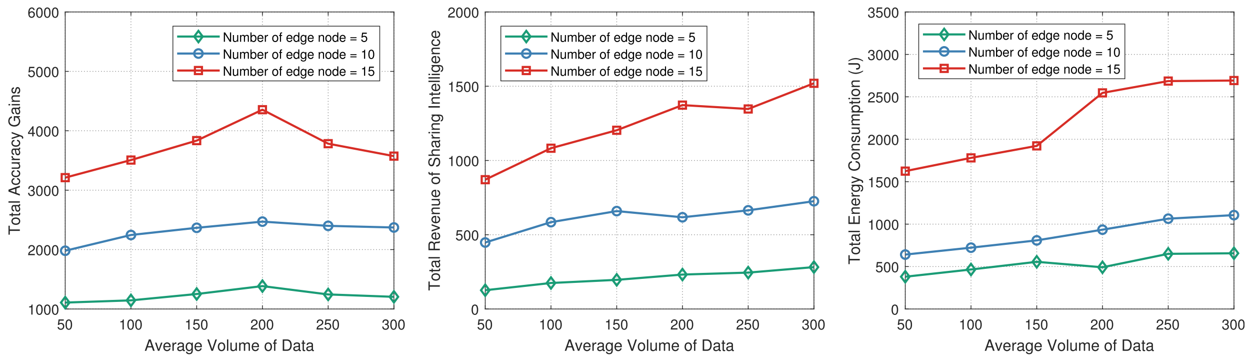
In Fig. 7, we test the performance of the proposed scheme, the SAC-based scheme and the DDPG-based scheme under different average computing capabilities of edge nodes. In particular, we analyze the performance of the three schemes under different metrics, such as total accuracy gain, total sharing revenue and total energy consumption. As shown in the figure, the total accuracy gain of edge nodes first increases with the increase of computing capability, and when the computing capability reaches 5 Gcycles, it decreases with the increase of computing capability. The total sharing revenue and total energy consumption of edge nodes increase as computing capability becomes stronger. This can be explained by the fact that the increase in the computing capability of edge nodes means that higher model accuracy can be obtained when training models. Meanwhile, stronger computing capability dramatically increases the energy consumption required to process each data sample. When the computing capability is small, the expansion speed of the model accuracy is greater than the rising speed of the energy consumption. Edge nodes will focus on the impact of model accuracy gain when making intelligence sharing decisions. However, as the computing capability is further expanded (for example, more than 5 Gcycles), the rising speed of the energy consumption may exceed the expansion speed of the model accuracy. In this case, the edge node may pay more attention to the impact of energy consumption, such as choosing a lower number of training iterations to reduce training energy consumption, resulting in a decrease in accuracy gain. Increased model accuracy and willingness to share intelligence will increase the revenue of sharing intelligence.

Last, we analyze the performance of the proposed scheme with different participating edge nodes under different average volumes of collected data in Fig. 8. Different metrics such as total accuracy gain, total sharing revenue and total energy consumption are used for evaluation. It



(a) Total Accuracy Gain (b) Total Sharing Revenue (c) Total Energy Consumption

Fig. 7. Performance of different intelligence sharing schemes for the given average computing capability.



(a) Total Accuracy Gain (b) Total Sharing Revenue (c) Total Energy Consumption

Fig. 8. Performance of the proposed scheme with different participating edge nodes for the given average volume of data.

can be observed that regardless of the number of participating edge nodes, the total accuracy gain of the proposed scheme always shows a trend of first increasing and then decreasing as the volume of data increases. The turning point is different under different participating edge nodes. Moreover, both the total sharing revenue and total energy consumption of edge nodes increase with the larger volume of data. The reason is as follows. The increased volume of data will directly affect the total accuracy gain and increase it, and more energy needs to be consumed to process each data sample. According to Eq. (5), as the volume of data increases, the increase rate of model accuracy gradually becomes slower. Therefore, when the volume of data is relatively small, the accuracy will have a higher impact on the total utility than the energy consumption, prompting the edge node to choose high-accuracy policies. As the volume of data continues to increase, its impact on accuracy gradually decreases, which makes edge nodes more inclined to choose intelligence sharing policies that can slow down the growth of energy consumption. The increase in the volume of data has improved the overall quality of models trained by the entire network, which has led to an upward trend in the total sharing revenue. The number of participating edge nodes also greatly affects the performance of the proposed scheme. The greater the number of participating edge nodes, the higher the values of the three performance metrics.

To sum up, from the above numerical results, we can see that the proposed scheme can effectively achieve convergence through continuous training. The proposed CDRL-based scheme has a faster convergence speed and better utility than the benchmark DRL-based schemes. Besides, our proposed scheme outperforms the benchmark schemes in terms of total utility, total accuracy gain, total sharing revenue and total energy consumption under different parameter settings. Compared with the benchmark intelligence sharing schemes, the proposed scheme can perform intelligence sharing more flexibly, thereby improving the efficiency of the edge network.

## 7 Conclusion

In this article, we investigated the distributed intelligence sharing problem in the Internet of intelligence-empowered edge computing to quickly and economically improve the performance of model training of the edge network through intelligence sharing and aggregation. Comprehensively considering the time-varying system states including the volume of collected data, the quality level of collected data, the computing capability of edge nodes, the channel conditions of edge networks and the reputation opinions between edge nodes, we formulated the distributed intelligence sharing as a multi-agent MDP problem. To tackle this problem efficiently and obtain the optimal intelligence sharing decision including training iteration selection, intelligence requesting and spectrum resource allocation, we designed a novel CDRL algorithm consisting of local SAC learning at each edge node and collective learning between different edge nodes. Finally, we conducted extensive experiments to validate the effectiveness and superiority of our proposed intelligence sharing scheme.

## References

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

[2] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep./Oct. 2019.

[3] X. Zhou et al., "A secure and privacy-preserving machine learning model sharing scheme for edge-enabled IoT," *IEEE Access*, vol. 9, pp. 17256–17265, 2021.

[4] J. Xie et al., "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 393–430, First Quarter 2019.

[5] B. Yang et al., "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 40–47, Apr. 2021.

[6] K. Zhang, Y. Zhu, S. Maharjan, and Y. Zhang, "Edge intelligence and blockchain empowered 5G beyond for the industrial Internet of Things," *IEEE Netw.*, vol. 33, no. 5, pp. 12–19, Sep./Oct. 2019.

[7] B. Yuan, J. Panneerselvam, L. Liu, N. Antonopoulos, and Y. Lu, "An inductive content-augmented network embedding model for edge artificial intelligence," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4295–4305, Jul. 2019.

[8] Y. An, F. R. Yu, J. Li, J. Chen, and V. C. M. Leung, "Edge intelligence (EI)-enabled HTTP anomaly detection framework for the Internet of Things (IoT)," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3554–3566, Mar. 2021.

[9] S. Liao et al., "Cognitive popularity based AI service sharing for software-defined information-centric networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2126–2136, Fourth Quarter 2020.

[10] Z. Zhou, S. Yang, L. Pu, and S. Yu, "CEFL: Online admission control, data scheduling, and accuracy tuning for cost-efficient federated learning across edge nodes," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9341–9356, Oct. 2020.

[11] X. Li, L. Cheng, C. Sun, K.-Y. Lam, X. Wang, and F. Li, "Federated-learning-empowered collaborative data sharing for vehicular edge networks," *IEEE Netw.*, vol. 35, no. 3, pp. 116–124, May/Jun. 2021.

[12] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2019, pp. 1–8.

[13] W. Wen et al., "Terngrad: Ternary gradients to reduce communication in distributed deep learning," 2017, *arXiv:1705.07878*.

[14] S. Wang et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[15] Q. Tang, F. R. Yu, R. Xie, A. Boukerche, T. Huang, and Y. Liu, "Internet of intelligence: A survey on the enabling technologies, applications, and challenges," *IEEE Commun. Surveys Tuts.*, early access, May 16, 2022, doi: 10.1109/COMST.2022.3175453.

[16] F. R. Yu, "From information networking to intelligence networking: Motivations, scenarios, and challenges," *IEEE Netw.*, vol. 35, no. 6, pp. 209–216, Nov./Dec. 2021.

[17] P. Lin, Q. Song, F. R. Yu, D. Wang, A. Jamalipour, and L. Guo, "Wireless virtual reality in beyond 5G systems with the internet of intelligence," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 70–77, Apr. 2021.

[18] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–7.

[19] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1188–1200, Feb. 2021.

[20] Y. Deng et al., "AUCTION: Automated and quality-aware client selection framework for efficient federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1996–2009, Aug. 2022.

[21] Y. Fu, F. R. Yu, C. Li, T. H. Luan, and Y. Zhang, "Vehicular blockchain-based collective learning for connected and autonomous vehicles," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 197–203, Apr. 2020.

[22] R. Xie, Q. Tang, C. Liang, F. R. Yu, and T. Huang, "Dynamic computation offloading in IoT fog systems with imperfect channel-state information: A POMDP approach," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 345–356, Jan. 2021.

[23] H. He, H. Shan, A. Huang, Q. Ye, and W. Zhuang, "Edge-aided computing and transmission scheduling for LTE-U-enabled IoT," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7881–7896, Dec. 2020.
[24] Q. Tang, R. Xie, T. Huang, W. Feng, and Y. Liu, "Dynamic computation offloading with imperfect state information in energy harvesting small cell networks: A partially observable stochastic game," *IEEE Wireless Commun. Lett.*, vol. 9, no. 8, pp. 1300–1304, Aug. 2020.
[25] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol*, vol. 68, no. 11, pp. 11 158–11 168, Nov. 2019.
[26] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Mar. 2021.
[27] M. Xu et al., "Multi-agent federated reinforcement learning for secure incentive mechanism in intelligent cyber-physical systems," *IEEE Internet Things J.*, early access, May 18, 2021, doi: 10.1109/JIOT.2021.3081626.
[28] W. Y. B. Lim et al., "Hierarchical incentive mechanism design for federated machine learning in mobile networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9575–9588, Oct. 2020.
[29] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1387–1395.
[30] N. Oren, T. J. Norman, and A. Preece, "Subjective logic and arguing with evidence," *Artif. Intell.*, vol. 171, no. 10/15, pp. 838–854, 2007.
[31] X. Huang, R. Yu, J. Kang, Z. Xia, and Y. Zhang, "Software defined networking for energy harvesting Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1389–1399, Jun. 2018.
[32] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10 700–10 714, Dec. 2019.
[33] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Trans. Veh. Technol*, vol. 68, no. 3, pp. 2906–2920, Mar. 2019.
[34] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, A. Manzoor, and C. S. Hong, "A crowdsourcing framework for on-device federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3241–3256, May 2020.
[35] Q. Ye, W. Shi, K. Qu, H. He, W. Zhuang, and X. Shen, "Joint RAN slicing and computation offloading for autonomous vehicular networks: A learning-assisted hierarchical approach," *IEEE Open J. Veh. Technol.*, vol. 2, pp. 272–288, 2021.
[36] W. Wu et al., "Dynamic RAN slicing for service-oriented vehicular networks via constrained learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2076–2089, Jul. 2021.
[37] W. Zhang et al., "Optimizing federated learning in distributed industrial IoT: A multi-agent approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3688–3703, Dec. 2021.
[38] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[39] W. Zhang et al., "Deep reinforcement learning based resource management for DNN inference in industrial IoT," *IEEE Trans. Veh. Technol*, vol. 70, no. 8, pp. 7605–7618, Aug. 2021.
[40] J. Achiam and S. Sastry, "Surprise-based intrinsic motivation for deep reinforcement learning," 2017, *arXiv:1703.01732*.
[41] X. Qiu, W. Zhang, W. Chen, and Z. Zheng, "Distributed and collective deep reinforcement learning for computation offloading: A practical perspective," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1085–1101, May 2021.
[42] P. Lin, Q. Song, F. R. Yu, D. Wang, and L. Guo, "Task offloading for wireless VR-enabled medical treatment with blockchain security using collective reinforcement learning," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15 749–15 761, Nov. 2021.
[43] N. C. Luong et al., "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, Oct.–Dec. 2019.
[44] X. Li, J. Wan, H.-N. Dai, M. Imran, M. Xia, and A. Celesti, "A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4225–4234, Jul. 2019.
[45] Y. Wang, Z. Su, T. Luan, R. Li, and K. Zhang, "Federated learning with fair incentives and robust aggregation for UAV-aided crowdsensing," *IEEE Trans. Netw. Sci. Eng.*, early access, Dec. 28, 2021, doi: 10.1109/TNSE.2021.3138928.
[46] Y. Zhan, P. Li, and S. Guo, "Experience-driven computational resource allocation of federated learning by deep reinforcement learning," in *Proc. IEEE Int. Parallel Distrib. Process.*, 2020, pp. 234–243.
[47] Q. Tang, R. Xie, F. R. Yu, T. Huang, and Y. Liu, "Decentralized computation offloading in IoT fog computing system with energy harvesting: A Dec-POMDP approach," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4898–4911, Jun. 2020.
[48] R. Zhang et al., "Buffer-aware virtual reality video streaming with personalized and private viewport prediction," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 2, pp. 694–709, Feb. 2021.
[49] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," *Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 8026–8037, 2019.
[50] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

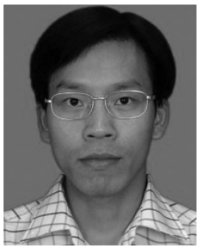
**Qinqin Tang** received the PhD degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2022. She is currently a postdoctoral research fellow with the Beijing University of Posts and Telecommunications. Her research interests include Internet of intelligence, edge/cloud computing, traffic offloading, and resource management.

**Renchao Xie** (Senior Member, IEEE) received the BS degree in electronic engineering from the Changchun University of Science and Technology, in 2007, and the MS and PhD degrees in electrical engineering from the Beijing University of Posts and Telecommunications, in 2009 and 2012, respectively. Currently, he is a Professor with the school of information and communication engineering with the Beijing University of Posts and Telecommunications. His research interests include edge computing, information centric networking, industrial internet of things. He has served as the Technical Program Committee (TPC) co-chair of numerous conferences. He has also served for several journals as a reviewer, including *IEEE Network*, *IEEE Trans. on Communications*, *IEEE Trans. On Vehicular Technology* and so on.

**Fei Richard Yu** (Fellow, IEEE) received the PhD degree in electrical engineering from the University of British Columbia (UBC), in 2003. His research interests include connected/autonomous vehicles, artificial intelligence, cybersecurity, and wireless systems. He has been named in the Clarivate Analytics list of "Highly Cited Researchers" since 2019, and received several best paper awards from some first-tier conferences. He is an elected member of the Board of Governors of the *IEEE Vehicular Technology Society* and editor-in-chief for *IEEE Vehicular Technology Society* Mobile World newsletter. He is a Fellow of the Canadian Academy of Engineering (CAE), Engineering Institute of Canada (EIC), and IET. He is a distinguished lecturer of IEEE in both VTS and ComSoc.

**Tianjiao Chen** received the PhD degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2022. He is currently a research fellow with China Mobile Research Institute. His research interests include machine learning, network virtualization and software defined networking (SDN).

**Ran Zhang** received the PhD degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2021. He is currently a postdoctoral research fellow with the Beijing University of Posts and Telecommunications. His research interests include caching, computing, and communication integration, named data networking, and artificial intelligence.



**Tao Huang** (Senior Member, IEEE) received the BS degree in communication engineering from Nankai University, Tianjin, China, in 2002, and the MS and PhD degree in communication and information system from the Beijing University of Posts and Telecommunications, Beijing, China, in 2004 and 2007, respectively. He is currently a Professor with the Beijing University of Posts and Telecommunications. His current research interests include network architecture, network artificial intelligence, routing and forwarding, and network virtualization.

**Yunjie Liu** received the BS degree in technical physics from Peking University, Beijing, China, in 1968. He is currently the academician of china academy of engineering, the chief of the Science and Technology Committee of China Unicom, and the dean of the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications. His current research interests include next generation network, and network architecture and management.