Optimizing Federated Learning in Distributed Industrial IoT: A Multi-Agent Approach

Weiting Zhang, *Student Member, IEEE*, Dong Yang, *Member, IEEE*, Wen Wu, *Member, IEEE*, Haixia Peng, *Member, IEEE*, Ning Zhang, *Senior Member, IEEE*, Hongke Zhang, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—In this paper, we aim to make the best joint decision of device selection and computing and spectrum resource allocation for optimizing federated learning (FL) performance in distributed industrial Internet of Things (IIoT) networks. To implement efficient FL over geographically dispersed data, we introduce a three-layer collaborative FL architecture to support deep neural network (DNN) training. Specifically, using the data dispersed in IIoT devices, the industrial gateways locally train the DNN model and the local models can be aggregated by their associated edge servers every FL epoch or by a cloud server every a few FL epochs for obtaining the global model. To optimally select participating devices and allocate computing and spectrum resources for training and transmitting the model parameters, we formulate a stochastic optimization problem with the objective of minimizing FL evaluating loss while satisfying delay and long-term energy consumption requirements. Since the objective function of the FL evaluating loss is implicit and the energy consumption is temporally correlated, it is difficult to solve the problem via traditional optimization methods. Thus, we propose a "*Reinforcement on Federated*" (RoF) scheme, based on deep multi-agent reinforcement learning, to solve the problem. Specifically, the RoF scheme is executed decentralizedly at edge servers, which can cooperatively make the optimal device selection and resource allocation decisions. Moreover, a device refinement subroutine is embedded into the RoF scheme to accelerate convergence while effectively saving the on-device energy. Simulation results demonstrate that the RoF scheme can facilitate efficient FL and achieve better performance compared with state-of-the-art benchmarks.**

***Index Terms*—Resource allocation, federated learning, industrial IoT, deep multi-agent reinforcement learning.**

Manuscript received March 2, 2021; revised July 26, 2021; accepted August 28, 2021. Date of publication October 7, 2021; date of current version November 22, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1702000, in part by the National Natural Science Foundation of China under Grant 61771040, and in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. An earlier version of this paper was presented in part at the IEEE ICC Workshop 2021 [DOI: 10.1109/ICCWorkshops50388.2021.9473515]. *(Corresponding author: Dong Yang.)*

Weiting Zhang, Dong Yang, and Hongke Zhang are with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: 17111018@bjtu.edu.cn; dyang@bjtu.edu.cn; hkzhang@bjtu.edu.cn).

Wen Wu and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 0B5, Canada (e-mail: w77wu@uwaterloo.ca; sshen@uwaterloo.ca).

Haixia Peng is with the Department of Computer Engineering and the Department of Computer Science, California State University Long Beach, Long Beach, CA 90840 USA (e-mail: haixia.peng@csulb.edu).

Ning Zhang is with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada (e-mail: ning.zhang@uwindsor.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2021.3118352.

Digital Object Identifier 10.1109/JSAC.2021.3118352

I. Introduction

With the wide deployment of industrial Internet of Things (IIoT), numerous devices are connected to the Internet, and voluminous industrial data is generated at the network edge [2]. As reported by Cisco, 2.4 TB data is burst out per minute from an industrial company in 2021 [3]. Utilized tremendous data, deep learning approaches, especially deep neural networks (DNNs) can facilitate industrial intelligence [4]. For example, a DNN model can be trained to provide fault diagnostic services for industrial facilities. In industrial scenarios, different factories usually generate a huge amount of data that contains critical information, such as the operation conditions of industrial facilities. Since this data information generally plays a crucial role in safety pre-warning and fault diagnostic, it is of great significance to preserve the privacy of industrial data. Traditional centralized training method needs to collect massive raw data from IIoT devices, e.g., industrial gateways (IGWs), which leads to user privacy concerns [5]. Hence, taking into account devices' privacy-preserving requirements, the data would be better processed locally [6]–[8].

Federated learning (FL), as an emerging distributed learning paradigm, can effectively solve the above issue [9]. Generally, FL operates over a two-layer architecture epoch-by-epoch. In each FL epoch, DNN models are locally trained at devices using their private dataset; and then the locally-trained model parameters are aggregated by a centralized server to update the global model parameters via a model aggregation algorithm, such as federated averaging (FedAvg) [10]. By running FL for multiple epochs until desired accuracy is achieved, a well-trained DNN model can be obtained to provide high-accuracy services. With the FL paradigm, DNN models can be efficiently trained based on the dispersed datasets held by devices without uploading raw data to a centralized server, thereby preserving privacy for the devices [11].

Very recently, advanced FL schemes have been widely applied in wireless edge networks [12]–[16]. In particular, applying FL in IIoT can greatly improve industrial data utilization with preserved data privacy. To this end, we focus on facilitating efficient FL in distributed IIoT networks. Smart factories of an industrial company are generally located in different geographical areas. To provide services for all factories, such as intelligent fault diagnostic, the company requires a global DNN model to achieve satisfactory performance over the data generated by these factories. However, due to excessive parameter transmission, traditional two-layer FL architecture will lead to high backbone communication overhead.

Accordingly, we introduce a three-layer architecture, i.e., device-edge-cloud, to support FL in the considered distributed scenario, in which the industrial data held by IGWs are geographically dispersed among factories. In each epoch, a set of IGWs need to be selected to participate in the FL, i.e., *device selection*. DNN model parameters that are locally trained at the selected IGWs are aggregated by edge servers every FL epoch, and the edge aggregated model parameters are transmitted to the cloud server for further aggregation every a few FL epochs. In this way, the amount of transmitted model parameters between the edge and the cloud can be greatly reduced. To efficiently support local training and parameter transmission, on-device computing and network spectrum resources need to be judiciously allocated for the participating devices, i.e., *resource allocation*. Due to limited network resources, selecting all devices to participate in each epoch may result in a prohibitive delay for parameter uploading, which in turn consumes excessively high on-device energy. Hence, device selection and resource allocation should be jointly considered to optimize FL with satisfied delay and energy consumption requirements.

Optimizing FL in the distributed and resource-limited IIoT networks encounters several challenges. *First*, in the FL, the optimization objective is to minimize the FL evaluating loss, which is represented by an implicit function. The mapping relationship between the objective function and decision variables cannot be characterized accurately. As such, it is difficult to directly solve the problem via traditional optimization methods. Also, the decision variables of the device selection and resource allocation are correlated, which further complicates decision making process. *Second*, the long-term energy consumption constraint couples to device selection and resource allocation decisions over time, and yet the decisions have to be made without foreseeing the future network dynamics. Inappropriate decision making may exhaust the on-device energy, consequently slowing down the FL convergence. To address the above challenges, deep reinforcement learning (RL) is a feasible approach to make prophetic decisions through capturing network dynamics [17]–[19].

In this work, we investigate how to leverage deep RL approaches to dynamically select the participating devices and allocate cherished spectrum and computing resources for optimizing FL in the distributed IIoT networks. First of all, we formulate a joint device selection and resource allocation (JDSRA) problem with the objective of minimizing the FL evaluating loss while satisfying the epoch delay and energy consumption requirements. This problem cannot be directly solved via traditional optimization methods due to the implicit loss function of FL and the temporal correlation of energy consumption constraint. Therefore, we propose a multi-agent learning-based resource management scheme to solve the formulated problem. The learning agents can judiciously make the optimal device selection and resource allocation decisions to efficiently support the long-term FL process without violating the long-term energy consumption and strict delay constraints. The maximum entropy term is incorporated into the reward function of the learning algorithm to effectively improve the exploration. Through training the multi-agent algorithm centrally, the learning agents, deployed at edge servers, can cooperatively make the optimal decisions. Extensive simulation results demonstrate that the proposed scheme achieves better FL performance, as compared to the benchmarks. The main contributions of this paper are summarized as follows:

- We introduce a three-layer collaborative FL architecture in the distributed IIoT networks, which can facilitate efficient FL while effectively reducing the backbone network traffic for DNN model parameter transmission;
- We formulate JDSRA as a stochastic optimization problem to optimize FL performance while satisfying the strict epoch delay and long-term energy consumption requirements, which is then transformed into a partially-observable Markov decision process (POMDP);
- We propose a "*Reinforcement on Federated*" (RoF) scheme, based on a deep multi-agent RL algorithm and a device refinement subroutine, to solve the POMDP. Through decentralized execution among multiple agents, the RoF scheme can cooperatively make the optimal decisions and accelerate convergence.

The remainder of this paper is organized as follows. Section II reviews related works. In Section III, a three-layer collaborative FL architecture is presented. Section IV presents the system model, and then the problem is formulated and transformed in Section V. Section VI proposes the learning-based RoF scheme. Simulation results are given in Section VII, and the concluding remarks are provided in Section VIII.

II. Related Work

Recently, there are several research works on FL. A line of works focus on device selection to accelerate FL convergence. Generally, selecting more devices in each FL epoch can improve the convergence rate. To this end, Yang *et al.* [20] aimed at maximizing the number of participating devices in each FL epoch via exploiting superposition property of wireless multiple-access channels. Nevertheless, a large number of participating devices leads to prohibitive communication cost due to excessive parameter transmission. Wang *et al.* [21] proposed a learning-based scheme to flexibly select a set of appropriate devices, consequently speeding up FL convergence. Different from the above works, our work focuses on cooperatively selecting devices for three-layer FL architecture via multiple learning agents. In addition, a device refinement subroutine is adopted to obtain a refined set of selected devices, which can ensure the FL can be completed with

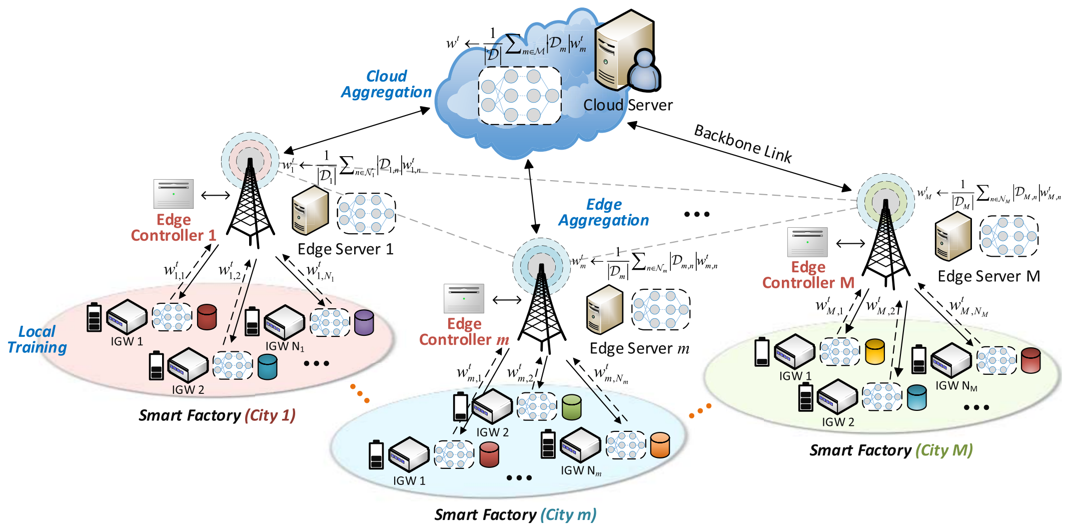


Fig. 1. The three-layer collaborative FL architecture for distributed IIoT networks.

satisfied delay and energy consumption requirements, thereby accelerating FL convergence.

Another line of works devote to resource allocation to facilitate efficient FL. In [11], an FL delay minimization problem is studied in cellular networks, in which the spectrum resource is dynamically allocated for parameter transmission. In [22], an opportunistic spectrum access scheme is proposed to minimize the energy consumption of FL. The above works focus on implementing FL based on a traditional two-layer architecture. Different from these works, our paper aims to optimize FL in a distributed IIoT network, in which the industrial data are dispersed geographically and backbone network is congested. In such case, instead of applying the two-layer architecture, we adopt a novel three-layer collaborative architecture to support FL, which can effectively reduce backbone network traffic for parameter transmission.

Deep RL techniques hold the potential to improve resource utilization in wireless networks via learning network dynamics [23]–[25]. Many advanced deep RL based algorithms have been developed to perform dynamic resource management. In [26], a deep Q-network (DQN) based algorithm is proposed to decide the devices to conduct data collection in an unmanned aerial vehicle-assisted mobile edge network. In [17], Chen *et al.* adopted a double DQN algorithm to make computation offloading and packet scheduling decisions in radio access networks. Peng and Shen [19] leveraged a deep deterministic policy gradient (DDPG) based algorithm to allocate spectrum resources in complex urban vehicular networks and learn the optimal task offloading policy. Moreover, aiming at optimizing the energy efficiency in a large-scale cognitive radio network, Kaur and Kumar [27] proposed a multi-agent Q-Learning based algorithm to obtain a decentralized radio resource allocation strategy. The above works provide valuable insights for achieving dynamic resource management in wireless networks. Different from the above works, we focus on facilitating efficient FL in distributed IIoT via leveraging deep RL approaches to judiciously allocate spectrum and computing resources.

In our preliminary paper [1], we proposed a centralized resource management scheme, in which a single-agent based algorithm is designed to globally allocate network resources. As an extension of [1], in this paper, we further propose a decentralized learning-based algorithm to perform judicious resource management for FL operation. In this algorithm, each geographically distributed base station (BS) is equipped with an edge controller that can only observe its individual environment information and make the decision for supporting FL independently.

III. Three-Layer Collaborative FL Architecture

To implement FL in distributed IIoT networks, we consider a three-layer collaborative FL architecture empowered with a hierarchical aggregation algorithm [28]. As shown in Fig. 1, the architecture is composed of the device, edge, and cloud layers endowed with different functionalities. The descriptions of each layer in the architecture are as follows.

- *Device Layer*: Energy-limited IGWs endowed with computing capability are geographically distributed in smart factories located in different cities. In each FL epoch, DNN models are locally trained at IGWs.
- *Edge Layer*: Multiple BSs are deployed to provide services for the IGWs. Each BS is equipped with an edge server to perform edge model parameter aggregation every FL epoch. Additionally, an edge controller deployed at each edge server can manage network resources and coordinate FL within the coverage of the edge server.
- *Cloud Layer*: A cloud server is deployed to perform global parameter aggregation, which is used to aggregate the updated parameters from edge servers after a few edge model parameter aggregations.

Within the three-layer architecture, DNN model parameters are infrequently transmitted between the edge and cloud,

Algorithm 1 Three-Layer FL Algorithm

```
1  ▷ Task initialization
2  Cloud: Initialize global DNN model parameter $w_0$;
3  ▷ Device selection and resource allocation
4  Obtain a refined set of participating IGWs $\mathcal{N}_m^\star, \forall m \in \mathcal{M}$;
5  for each $FL\ epoch\ t \in \mathcal{R}$ do
6      ▷ Model parameter distribution (downlink spectrum)
7      Broadcast $w_m$ to the selected IGWs;
8      ▷ Local model training (on-device computing)
9      for each $BS\ m \in \mathcal{M}$ in parallel do
10         for each $IGW\ n \in \mathcal{N}_m^\star$ in parallel do
11             for each training sample $d \in \mathcal{D}_{m,n}^T$ do
12                 $w_{m,n} \leftarrow w_m - \lambda \nabla f(x_d, y_d; w_{m,n})$;
13     ▷ Edge aggregation (uplink spectrum)
14     if $mod(t, t_c) \neq 0$ then
15         $w_m \leftarrow 1/\left|\mathcal{D}_m^E\right| \sum_{m \in \mathcal{M}, n \in \mathcal{N}_m^\star} \left|\mathcal{D}_{m,n}^E\right| w_{m,n}$;
16         $F(w) = F_e(w_m)$;
17     ▷ Cloud aggregation (uplink spectrum)
18     if $mod(t, t_c) = 0$ then
19         $w \leftarrow 1/\left|\mathcal{D}^E\right| \sum_{m \in \mathcal{M}} \left|\mathcal{D}_m^E\right| w_m$;
20         $F(w) = F_c(w)$;
21 return $w$, $F(w)$.
```

such that the backbone network traffic can be effectively reduced.[1]

In this architecture, we consider a set of edge servers, denoted by $\mathcal{M} = \{1, 2, \cdots, M\}$. Let $\mathcal{N}_m = \{1, 2, \cdots, |\mathcal{N}_m|\}$ be the set of IGWs within the coverage of edge server $m \in \mathcal{M}$. Let $\mathcal{D}_{m,n} = \{(x_d, y_d)\}_d$ denote the local data set held by IGW n under edge server m, where x_d and y_d are the input and the label of each sample, respectively, and d is the sample index. Moreover, $\mathcal{D}_{m,n}^T$ and $\mathcal{D}_{m,n}^E$ are the training and evaluating data sets of the IGW, respectively, where $\mathcal{D}_{m,n}^T \cup \mathcal{D}_{m,n}^E = \mathcal{D}_{m,n}$. Note that the training data samples are assumed to follow a non-independent and identical distribution (non-IID). Similarly, $\mathcal{D}_m^T$ and $\mathcal{D}_m^E$ are the total training and evaluating data sets of the participating devices covered by edge server m, respectively, where $\mathcal{D}_m^T \cup \mathcal{D}_m^E = \mathcal{D}_m$. Notably, due to the data privacy preserving requirement, the data samples held by each IGW cannot be shared with others during the FL process.

The operation procedure of the FL algorithm based on the three-layer architecture is given in Algorithm 1, which is detailed as follows.

- *Task Initialization (Lines 1-2)*: In the initialization stage, a training task is initiated at the cloud server, and a global DNN model with a set of initialized parameters w_0 is instantiated, such as a diagnostic DNN model that is utilized to classify the health conditions of industrial equipments.
- *Device Selection and Resource Allocation (Lines 3-4)*: Once the training task and the global DNN model are initiated, the edge controllers will make the device selection and resource allocation decisions for FL epoch $t \in \mathcal{R}$, respectively. In specific, the decisions include: (1) the set of IGWs that are selected to participate in the current FL epoch; and (2) the amounts of downlink/uplink spectrum and computing resources that are allocated to the selected IGWs. Given the decisions, a refined device set $\mathcal{N}_m^\star, \forall m \in \mathcal{M}$ can be obtained accordingly.
- *Global Model Parameter Distribution (Lines 6-7)*: With the refined device set $\mathcal{N}_m^\star, \forall m \in \mathcal{M}$, the global model parameters can be distributed from the edge or cloud to the participating IGWs.
- *Local Model Training (Lines 8-12)*: Once the global model parameters w_m are received, the participating IGWs will respectively perform model training on their local datasets for one FL epoch. Here, the parameters are updated via a stochastic gradient descend method, i.e.,

$$w_{m,n} = w_m - \lambda \nabla f(x_d, y_d; w_{m,n}), \quad \forall d \in \mathcal{D}_{m,n}^T, \tag{1}$$

where λ is the learning rate, $w_{m,n}$ is the updated parameters of each participating IGW, and $f(x_d, y_d; w_{m,n})$ is the loss on each training sample of the IGWs.
- *Edge Aggregation (Lines 13-16)*: When the local update is completed, each edge server synchronously aggregates model parameters from the participating IGWs at each FL epoch, i.e., $mod(t, t_c) \neq 0$, and updates its global model parameter via the FedAvg algorithm, namely,

$$w_m = \frac{1}{|\mathcal{D}_m^E|} \sum_{m \in \mathcal{M}, n \in \mathcal{N}_m^\star} \left|\mathcal{D}_{m,n}^E\right| w_{m,n}, \tag{2}$$

where $\left|\mathcal{D}_{m,n}^E\right|$ and $\left|\mathcal{D}_m^E\right|$ are the numbers of evaluating samples held by each participating IGW and by the participating IGWs of edge server m. In the next FL epoch, the aggregated model parameters on each edge server can be distributed to the newly participating IGWs for further update. In such case, the evaluating loss of the DNN model can be described as

$$F_e(w_m) = \frac{1}{M} \sum_{m \in \mathcal{M}} F_m(w_m), \tag{3}$$

where

$$F_m(w_m) = \frac{1}{|\mathcal{D}_m^E|} \sum_{d \in \mathcal{D}_m^E} f(x_d, y_d; w_m) + \frac{\mu}{2}\|w_m\|^2, \tag{4}$$

where $f(x_d, y_d; w_m)$ denotes the loss on each evaluating sample of the participating IGWs with parameters w_m aggregated by edge server m, and $\mu\|w_m\|^2/2$ is the regularization term that is added to prevent parameter overfitting, and μ is the regularization coefficient.
- *Cloud Aggregation (Lines 17-20)*: The cloud server aggregates model parameters from edge servers every t_c FL epochs, i.e., when $mod(t, t_c) = 0$, and updates the global model via the FedAvg algorithm, i.e.,

$$w = \frac{1}{|\mathcal{D}^E|} \sum_{m \in \mathcal{M}} \left|\mathcal{D}_m^E\right| w_m. \tag{5}$$

[1]The proposed three-layer architecture can also work in the case that not all the IGWs are able to connect the edge server. In such case, the locally trained model parameters can be directly transmitted to the cloud server for the global aggregation.

The evaluating loss is given by

$$F_c(\boldsymbol{w}) = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} f(\boldsymbol{x}_d, y_d; \boldsymbol{w}) + \frac{\mu}{2}\|\boldsymbol{w}\|^2, \tag{6}$$

where $f(\boldsymbol{x}_d, y_d; \boldsymbol{w})$ represents the evaluating loss on each evaluating sample of the participating IGWs with parameters $\boldsymbol{w}$ aggregated by the cloud server, and $|\mathcal{D}|$ denotes the total number of evaluating samples held by all participating IGWs in the current FL epoch.

Note that when the connection between an IGW and the corresponding edge server is established, the IGW can send its information, including the number of training samples held by the IGW, to the edge server, and then such information is further reported to the centralized node (i.e., the cloud server). As such, the cloud server can have the information of the numbers of training samples held by distributed IGWs.

IV. System Model

In this section, we present the FL delay and energy consumption models. Then, the FL evaluating loss is defined to measure the FL performance.

A. FL Delay Model

As previously mentioned, the FL process operates epoch-by-epoch. To facilitate efficient FL, each epoch is constrained by a delay requirement $T^{\max}$, and the total delay of each IGW during the FL operation is constrained by a budget $T^{\max}_{m,n}$. In specific, the delay for FL includes three parts, which are defined as follows.

1) Delay of Global Model Distribution: In FL epoch t, a certain number of IGWs needs to be selected to participate in the FL. Let $o_{m,n,t} \in \{0,1\}, \forall m \in \mathcal{M}, n \in \mathcal{N}_m$ be the binary decision variable of device selection. Specifically, if $o_{m,n,t}$ equals 1, IGW n under edge server m is then selected, otherwise 0. We adopt orthogonal frequency allocation to support model parameter distribution and aggregation, which are operated with the available bandwidth $B^{\downarrow}$ (downlink) and $B^{\uparrow}$ (uplink), respectively. Here, the underlying assumption is that we do not consider other traffic loads in the considered scenario. This assumption is widely adopted in performance analysis of FL frameworks [8].

Thus, the downlink spectrum efficiency for edge server m is given by $\gamma^{\downarrow}_{m,n,t} = \log_2\left(1 + p_{m,t}|g_{m,n,t}|^2/\sigma^2\right), \forall m \in \mathcal{M}, n \in \mathcal{N}_m$, where $p_{m,t}$ is the transmission power, $|g_{m,n,t}|^2$ is the channel gain between edge server m and IGW n, and σ^2 is the Gaussian noise power [29]. A fraction $\xi^{\downarrow}_{m,n,t}$ of bandwidth $B^{\downarrow}$ is allocated to IGW n from edge server m. Here, we have $0 \le \xi^{\downarrow}_{m,n,t} \le 1$, and $\sum_{m\in\mathcal{M},n\in\mathcal{N}_m} o_{m,n,t}\xi^{\downarrow}_{m,n,t} = 1$. Note that this paper focuses on the spectrum allocation among IIoT devices, i.e., what percentage of resources should be allocated to an IIoT device. As such, we adopt a fractional bandwidth model. Hence, the downlink transmission rate of edge server m for IGW n is $R^{\downarrow}_{m,n,t} = \xi^{\downarrow}_{m,n,t}B^{\downarrow}\gamma^{\downarrow}_{m,n,t}, \forall m \in \mathcal{M}, n \in \mathcal{N}_m$.

When distributing model parameters to IGW n, the transmission delay $d^{\downarrow}_{m,n,t}$ can be calculated based on the following two cases: (1) $d^{\downarrow}_{m,n,t} = S_g/R^{\downarrow}_{m,n,t} + S_g/R^c$, when the global model parameters are distributed from the cloud server, where S_g (in bits) is the data size of the global DNN model parameter and R^c represents the backhaul transmission rate between the cloud and the edge; and (2) $d^{\downarrow}_{m,n,t} = S_g/R^{\downarrow}_{m,n,t}$, when the aggregated model parameters are distributed from edge servers to its covered IGWs.

With the orthogonal frequency allocation mechanism, the delay for distributing model parameters is determined by the slowest one, namely,

$$T^{\downarrow}_{t,com} = \max_{m\in\mathcal{M},n\in\mathcal{N}_m} d^{\downarrow}_{m,n,t}, \tag{7}$$

where $\max\{\cdot\}$ is the function to return the largest value.

2) Delay of Local Model Training: In the considered IIoT networks, the distributed IGWs are endowed with a certain on-device computing capabilities, denoted by f (central processing unit (CPU) cycles per second). The IGW's computing resource should be judiciously allocated to ensure that local model training can be completed within the strict delay threshold. Let $\eta_{m,n,t}$ be the decision variable of the fraction of computing resource the IGW used in FL epoch t, where $0 \le \eta_{m,n,t} \le 1$. Thus, the delay for local model training can be given by

$$\tau_{m,n,t} = \frac{a\left|\mathcal{D}^T_{m,n}\right|}{\eta_{m,n,t}f}, \quad \forall m \in \mathcal{M},\ n \in \mathcal{N}_m, \tag{8}$$

where a is the number of CPU cycles required for processing per bit data, and $\left|\mathcal{D}^T_{m,n}\right|$ is the size of training set held by each participating device. For different FL models, parameter a is set accordingly to reflect the effect of the adopted models on the training delay and energy consumption. Similarly, the delay for local model training is determined by the slowest one, i.e.,

$$T_{t,cmp} = \max_{m\in\mathcal{M},n\in\mathcal{N}_m} \tau_{m,n,t}. \tag{9}$$

3) Delay of Model Parameter Uploading: Similar to the model parameter distribution, the uplink spectrum efficiency for parameter aggregation can be expressed as $\gamma^{\uparrow}_{m,n,t} = \log_2\left(1 + p_{m,n,t}|g_{m,n,t}|^2/\sigma^2\right), \forall m \in \mathcal{M}, n \in \mathcal{N}_m$, where $p_{m,n,t}$ is the transmission power from IGW n to edge server m [30]. Let $\xi^{\uparrow}_{m,n,t}(0 \le \xi^{\uparrow}_{m,n,t} \le 1)$ denotes a decision variable of the fraction of uplink bandwidth $B^{\uparrow}$ allocated to IGW n, where $\sum_{m\in\mathcal{M},n\in\mathcal{N}_m} o_{m,n,t}\xi^{\uparrow}_{m,n,t} = 1$. Hence, the uplink transmission rate can be described as $R^{\uparrow}_{m,n,t} = \xi^{\uparrow}_{m,n,t}B^{\uparrow}\gamma^{\uparrow}_{m,n,t}, \forall m \in \mathcal{M}, n \in \mathcal{N}_m$. The transmission delay $d^{\uparrow}_{m,n,t}$ of uploading DNN model parameters from IGW n to edge server m also has two cases: (1) for the edge aggregation, the updated model parameters of IGWs need to be aggregated by the correlated edge server, and $d^{\uparrow}_{m,n,t} = S_l/R^{\uparrow}_{m,n,t}$; and (2) for the cloud aggregation, $d^{\uparrow}_{m,n,t} = S_l/R^{\uparrow}_{m,n,t} + S_l/R^c$. Here, S_l (in bits) is the parameter size of the local DNN model. Since the model parameter size is the same during the FL operation, we have $S_l = S_g$. In addition, DNN model parameters are infrequently transmitted between the edge and the cloud, thus we assume the backhaul transmission rates R^c in uplink and downlink are the same. Due to the synchronous aggregation mechanism, the delay for distributing model parameters is determined by the slowest IGW, i.e.,

$$T^{\uparrow}_{t,com} = \max_{m\in\mathcal{M},n\in\mathcal{N}_m} d^{\uparrow}_{m,n,t}. \tag{10}$$

Taking $T^{\downarrow}_{t,com}$, $T_{t,cmp}$, and $T^{\uparrow}_{t,com}$ into consideration, the total delay for each FL epoch can be given by

$$T_t = T^{\downarrow}_{t,com} + T_{t,cmp} + T^{\uparrow}_{t,com}. \tag{11}$$

In addition, given the model parameter distribution, local training, and parameter uploading delays, the cumulative FL delay of each IGW is described as

$$T_{m,n,t} = T_{m,n,t-1} + d^{\downarrow}_{m,n,t} + \tau_{m,n,t} + d^{\uparrow}_{m,n,t}, \tag{12}$$

which is used to calculate the remaining time of FL, and thus affecting whether the IGWs can participate in the subsequent FL epochs. Here, $T_{m,n,t-1}$ denotes the cumulative FL delay of each IGW until epoch $t-1$.

B. Energy Consumption Model

In the long-term FL process, IGWs are constrained by their own energy capacity $E^{\max}_{m,n}$, which should be satisfied if the IGW is selected to participate in the FL. In specific, the energy consumption for FL includes two parts, which are calculated as follows, respectively.

1) Energy Consumption of Parameter Uploading: In the model parameter uploading stage, the energy consumption of IGW n covered by edge server m is given by

$$E^{com}_{m,n,t} = p_{m,n,t} d^{\uparrow}_{m,n,t}, \quad \forall m \in \mathcal{M},\ n \in \mathcal{N}_m. \tag{13}$$

2) Energy Consumption of Local Training: Performing local model training at IGWs also consumes energy, which is affected by the allocated computing resource of the participating IGWs and the computing workload of the local training process. Hence, the energy consumption of local training at each IGW can be given by

$$E^{cmp}_{m,n,t} = \frac{\kappa}{2} a \left|\mathcal{D}^{T}_{m,n}\right| (\eta_{m,n,t} f)^2, \quad \forall m \in \mathcal{M},\ n \in \mathcal{N}_m, \tag{14}$$

where κ is the effective capacitance coefficient of the computing chipset of IGWs [28].

Taking the above two parts into consideration, the cumulative energy consumption can be calculated by

$$E_{m,n,t} = E_{m,n,t-1} + E^{com}_{m,n,t} + E^{cmp}_{m,n,t}, \quad \forall m \in \mathcal{M}, n \in \mathcal{N}_m, \tag{15}$$

which is utilized to calculate the remaining energy resource of IGWs, and hence evaluating whether the IGWs satisfy the long-term energy consumption requirements. Here, $E_{m,n,t-1}$ denotes the cumulative energy consumption of each IGW until epoch $t-1$.

C. FL Evaluating Loss

In the considered architecture, FL operates epoch-by-epoch with the selected IGWs, which can satisfy the FL epoch delay requirement $T^{\max}$, time budget $T^{\max}_{m,n}$, and energy consumption constraint $E^{\max}_{m,n}$. The iteration process aims to obtain the optimal model parameters $\boldsymbol{w}^*$ that can minimize the evaluating loss on the evaluating datasets of the participating IGWs. To optimize FL under both the edge and cloud aggregation cases, the FL evaluating loss can be defined as

$$F(\boldsymbol{w}) = \begin{cases} F_e(\boldsymbol{w}_m), & \text{if } mod(t, t_c) \neq 0; \\ F_c(\boldsymbol{w}), & \text{otherwise.} \end{cases} \tag{16}$$

If $mod(t, t_c) \neq 0$, edge servers aggregate the covered model parameters from the participating IGWs, respectively, and obtain the evaluating loss $F_e(\boldsymbol{w}_m)$ with the aggregated parameters $\boldsymbol{w}_m, \forall m \in \mathcal{M}$; otherwise, the cloud server aggregates the model parameters from edge servers and obtains the evaluating loss $F_c(\boldsymbol{w})$ with the global parameters $\boldsymbol{w}$.

V. Problem Formulation and Transformation

In this section, we formulate an optimization problem to manage the spectrum and computing resources and then transform the formulated problem into a POMDP.

A. Problem Formulation

As previously described, the FL evaluating loss $F(\boldsymbol{w})$ is used to optimize the performance in the FL process. Thus, to minimize $F(\boldsymbol{w})$ over the aggregated model parameters and IGWs' private datasets while satisfying the FL epoch delay, time budget, and IGWs' long-term energy consumption requirements, we formulate the problem as follows:

$$\mathbf{P}_0: \min_{\boldsymbol{o},\boldsymbol{\xi}^{\downarrow},\boldsymbol{\xi}^{\uparrow},\boldsymbol{\eta}} F(\boldsymbol{w})$$

$$\text{s.t.} \quad \mathrm{E_{m,n,t} \leq E^{max}_{m,n}}, \quad \forall \mathrm{m} \in \mathcal{M},\ \mathrm{n} \in \mathcal{N}_\mathrm{m},\ \mathrm{t} \in \mathcal{R}, \tag{17a}$$

$$T_{m,n,t} \leq T^{\max}_{m,n}, \quad \forall m \in \mathcal{M}, n \in \mathcal{N}_m,\ t \in \mathcal{R}, \tag{17b}$$

$$T_t \leq T^{\max}, \quad \forall t \in \mathcal{R}, \tag{17c}$$

$$o_{m,n,t} \in \{0,1\}, \quad \forall m \in \mathcal{M}, n \in \mathcal{N}_m,\ t \in \mathcal{R}, \tag{17d}$$

$$\sum_{m\in\mathcal{M},n\in\mathcal{N}_m} o_{m,n,t}\xi^{\downarrow}_{m,n,t} = 1, 0 \leq \xi^{\downarrow}_{m,n,t} \leq 1, \tag{17e}$$

$$\sum_{m\in\mathcal{M},n\in\mathcal{N}_m} o_{m,n,t}\xi^{\uparrow}_{m,n,t} = 1, 0 \leq \xi^{\uparrow}_{m,n,t} \leq 1, \tag{17f}$$

$$0 \leq \eta_{m,n,t} \leq 1, \quad \forall m \in \mathcal{M}, n \in \mathcal{N}_m,\ t \in \mathcal{R}, \tag{17g}$$

where $\boldsymbol{o} = \{o_{m,n,t}\}_{\forall m\in\mathcal{M},n\in\mathcal{N}_m}$, $\boldsymbol{\xi}^{\downarrow} = \{\xi^{\downarrow}_{m,n,t}\}_{\forall m\in\mathcal{M},n\in\mathcal{N}_m}$, $\boldsymbol{\xi}^{\uparrow} = \{\xi^{\uparrow}_{m,n,t}\}_{\forall m\in\mathcal{M},n\in\mathcal{N}_m}$, and $\boldsymbol{\eta} = \{\eta_{m,n,t}\}_{\forall m\in\mathcal{M},n\in\mathcal{N}_m}$. With these decisions, the model parameters $\boldsymbol{w}$ are optimized in the local model training stage via optimization methods, such as the stochastic gradient decent method [31]. Constraints (17a), (17b), and (17c) ensure the cumulative energy consumption and time, and FL epoch delay can be restricted within $E^{\max}_{m,n}$, $T^{\max}_{m,n}$, and $T^{\max}$, respectively.

Note that although a synchronous mechanism is adopted to aggregate the distributed model parameters, the cumulative delay for each IGW is different since the IGWs would go to sleep after the stage of parameter distribution, local training, or parameter uploading. Thus, the practical service delay in each epoch is different during the FL operation, and the constraint (17b) ensures the cumulative delay for each IGW can be restricted within $T_{m,n}^{max}$. In addition, inappropriate resource allocation decisions may lead to a large delay for parameter distribution, local training, and parameter uploading, and therefore reducing the effectiveness of the FL operation. Thus, the constraint (17c) is adopted to ensure the epoch delay can be restricted within T^{max}. Constraint (17d) indicates whether the corresponding IGW is selected to participate in FL epoch t. Constraints (17e)-(17g) guarantee the available bandwidth $B^{\downarrow}$ and $B^{\uparrow}$, and on-device computing resource f can be judiciously allocated.

Through solving the formulated loss minimization problem, FL performance can be efficiently optimized in the considered distributed IIoT networks with satisfied multiple requirements. However, there are some challenges in solving the problem.

- The objective function of problem $\mathbf{P}_0$ is implicit and constraint (17a) is temporally correlated, which prevent traditional optimization methods to solve the problem. Due to the strict delay requirement of FL epochs, problem $\mathbf{P}_0$ should be solved with a low latency.
- Problem $\mathbf{P}_0$ is a mix-integer nonlinear optimization problem, which contains both discrete and continuous decision variables. Also, the considered decision variables are correlated with each other, which further complicates decision making process.

The device selection and resource allocation aim at minimizing the long-term evaluating loss, which can be reformulated as a Markov decision process. To obtain the optimal solution, deep RL is an effective approach endowed with strong representation capabilities, which can effectively solve the resource optimization problem with mixed decision variables and temporally correlated constraints [32], [33]. However, single-agent based deep RL algorithms generally require a global controller to collect all the state information from distributed IGWs and observe the entire network environment, which incurs backbone signaling overhead. To address the issue, a deep multi-agent RL algorithm should be adopted, in which each edge controller acts as an agent to cooperatively learn a dynamic resource management policy and solve the problem.

B. Problem Transformation

To optimize FL in the distributed IIoT networks, we first transform the formulated problem $\mathbf{P}_0$ into a POMDP for M agents [34]. In the considered distributed scenario, the edge controllers are modelled as the *edge agents*, respectively. Each edge agent can observe the state information from their individual environment. For the transformed POMDP, the edge agents can capture the dynamics, i.e., the remaining energy resource of IGWs, from the environment to learn their optimal policies $\{\boldsymbol{\pi}^*_{\boldsymbol{\theta}_m}\}_{\forall m \in \mathcal{M}}$ parameterized by $\boldsymbol{\theta}_m$. The policies can accurately map a set of states $\{\boldsymbol{s}_{m,t}\}_{\forall m \in \mathcal{M}}$ to a set of actions $\{\boldsymbol{a}_{m,t}\}_{\forall m \in \mathcal{M}}$, which is evaluated by the reward function $\{\boldsymbol{r}_{m,t}\}_{\forall m \in \mathcal{M}}$. The state, action, and reward can be defined as follows.

1) State: In FL epoch t, each edge agent collects the state information from their individual environment, which indicates the remaining FL time and energy of IGWs. Namely,

$$\boldsymbol{s}_{m,t} = \{T^r_{m,n,t}, E^r_{m,n,t}\}, \quad \forall m \in \mathcal{M},\ n \in \mathcal{N}_m, \tag{18}$$

where $T^r_{m,n,t} = T^{\max}_{m,n} - T_{m,n,t-1}$ and $E^r_{m,n,t} = E^{\max}_{m,n} - E_{m,n,t-1}$.

2) Action: According to the observed individual environment states, the edge agents cooperatively make decisions to select the IGWs to participate in the current FL epoch, and allocate the downlink/uplink spectrum and on-device computing resources for supporting the FL, i.e.,

$$\boldsymbol{a}_{m,t} = \{o_{m,n,t}, \xi^{\downarrow}_{m,n,t}, \xi^{\uparrow}_{m,n,t}, \eta_{m,n,t}\}, \quad \forall m \in \mathcal{M},\ n \in \mathcal{N}_m. \tag{19}$$

Each action element is reshaped into a range of $[0, 1]$. Note that only when $o_{m,n,t}$ equals 1, the IGW then can use the allocated spectrum and computing resources to support FL, such that constraints (17d)-(17g) can be satisfied.

3) Reward: Once the action $\boldsymbol{a}_{m,t}$ of edge agent m has been determined based on the observed individual state $\boldsymbol{s}_{m,t}$. The agent will obtain a reward to evaluate the quality of the action. To minimize the FL evaluating loss, the reward function of the multi-agent learning algorithm then can be defined as

$$\boldsymbol{r}_{m,t} = \begin{cases} -F_e(\boldsymbol{w}_m), & \text{if } \textit{mod}(t, t_c) \neq 0, \\ -F_c(\boldsymbol{w}), & \text{if } \textit{mod}(t, t_c) = 0, \\ -U, & \text{if no aggregation,} \end{cases} \tag{20}$$

where $F_e(\boldsymbol{w}_m)$ and $F_c(\boldsymbol{w})$ are the FL evaluating losses of DNN models after the edge and cloud aggregations, respectively. Considering the agents are cooperatively performing the FL operation, $F_e(\boldsymbol{w}_m)$ and $F_c(\boldsymbol{w})$ are set as the rewards for each agent in these two aggregation cases. Note that $F_e(\boldsymbol{w}_m)$ is obtained by a mean value of $F_m(\boldsymbol{w}_m)$ that is defined in Eq. (4). An edge at the edge can obtain the mean loss via the following steps. First, the loss value $F_m(\boldsymbol{w}_m)$ evaluated at the edge servers is transmitted to a cloud server. Second, the mean loss, $F_e(\boldsymbol{w}_m) = \frac{1}{M}\sum_{m \in \mathcal{M}} F_m(\boldsymbol{w}_m)$, is calculated by the cloud server. Third, $F_e(\boldsymbol{w}_m)$ is broadcasted to each edge server, which is used as the reward of the edge agent. In addition, U is a penalty factor, which should take a relatively large value to penalize the decisions that cannot aggregate any model parameters within constraints (17a)-(17c) in each FL epoch. In particular, the minus before the above three terms is mainly used to transform the original loss minimization problem into a reward maximization problem.

In the POMDP, each agent m aims at maximizing its own cumulative discounted reward with the optimal device selection and resource allocation policy $\boldsymbol{\pi}^*_{\boldsymbol{\theta}_m}$. As such, $\mathbf{P}_0$ can be reformulated as multiple distributed optimization subproblems, i.e.,

$$\mathbf{P}_{m,1}: \quad \max_{\boldsymbol{\pi}_{\boldsymbol{\theta}_m}} \mathbb{E}\left[\sum_{\delta=0}^{\infty} \gamma^{\delta} r_{m,t+\delta} | \boldsymbol{\pi}_{\boldsymbol{\theta}_m}, \boldsymbol{s}_{m,t}, \boldsymbol{a}_{m,t}\right]$$
$$\text{s.t.} \ \ (17\text{a}) - (17\text{g}), \tag{21a}$$

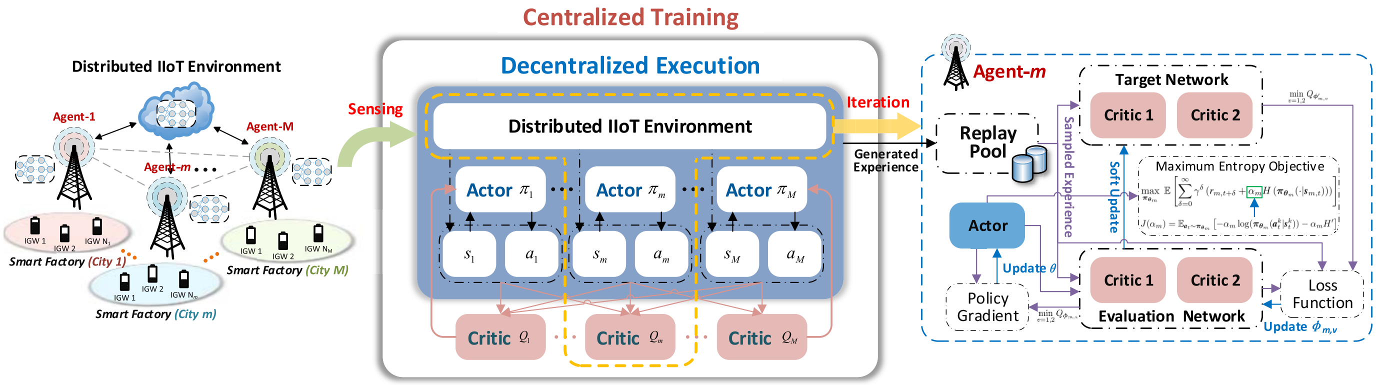


Fig. 2. The architecture of the proposed MASAC-based resource management algorithm.

where $\gamma \in [0, 1]$ is the discount factor. Due to the cooperative relationship among the edge agents, sub-problems $\mathbf{P}_{m,1} (\forall m \in \mathcal{M})$ are correlated with each other.

VI. RoF: A Multi-Agent Scheme for Efficient FL

Taking into account the correlation among sub-problems $\mathbf{P}_{m,1} (\forall m \in \mathcal{M})$, which should be jointly solved to achieve the same goal, i.e., reward maximization. Moreover, due to the mix-integer decision variables, instead of value-based RL algorithms, we adopt a policy gradient-based algorithm, i.e., soft actor-critic (SAC) [35], for each agent to address its POMDP with more robust learning performance.

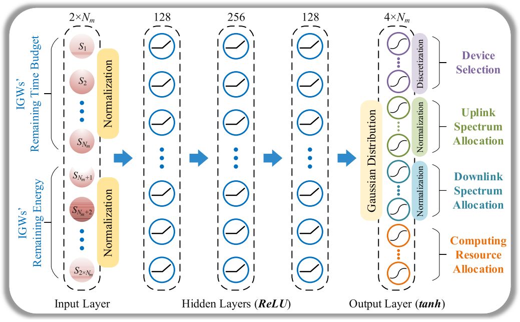
In this section, we propose the RoF scheme, based on a multi-agent SAC (MASAC)-based resource management algorithm and a device refinement subroutine, to cooperatively solve the modelled POMDP for M edge agents. As illustrated in Fig. 2, the proposed MASAC-based algorithm that combines the SAC algorithm with a cooperative multi-agent RL framework [36], is presented. In the following, we first introduce the SAC and multi-agent RL framework, and then present the MASAC and the device refinement subroutine in detail, followed by the computational complexity analysis of the MASAC.

A. SAC and Multi-Agent RL Framework

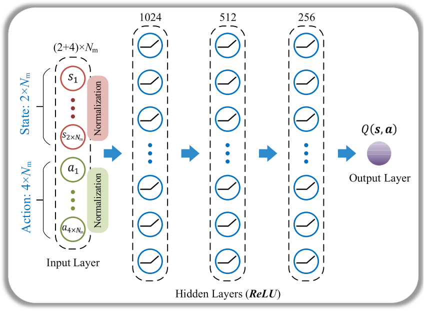
SAC Algorithm: As shown in the right part of Fig. 2, each edge agent is based on the SAC algorithm. In the SAC, three main components are incorporated to enhance the performance: (1) entropy maximization, (2) "actor-critic" network architecture, and (3) off-policy formulation, which are detailed as follows.

1) Maximum entropy is considered to effectively enhance the learning stability and exploration, which aims to maximize both the cumulative discounted reward and the expected entropy of the policy simultaneously. With this maximum entropy objective, the stochasticity of the agent's policy can be greatly improved while enabling more possible optimal decisions to be explored. As such, the reformulated sub-problems $\mathbf{P}_{m,1} (\forall m \in \mathcal{M})$ can be rewritten as

$$\mathbf{P}_{m,2}: \max_{\boldsymbol{\pi}_{\boldsymbol{\theta}_m}} \mathbb{E}\left[\sum_{\delta=0}^{\infty} \gamma^{\delta} \left(r_{m,t+\delta} + \alpha_m H\left(\boldsymbol{\pi}_{\boldsymbol{\theta}_m}(\cdot|\boldsymbol{s}_{m,t})\right)\right)\right]$$
$$\text{s.t. } (17a) - (17g), \qquad (22a)$$



(a) Actor network



(b) Critic network

Fig. 3. Actor and critic network structures of MASAC.

where $H(\cdot)$ is the entropy term, and α_m is a weight that controls the relative importance of both terms.

2) Actor-critic architecture of each edge agent is composed of an actor network, a pair of evaluation critic networks, and a pair of target critic networks. The actor makes decisions $\boldsymbol{a}_{m,t}$ with its own policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_m}$. The critics calculate a pair of Q-values for evaluating the policy, respectively. As shown in Fig. 3, the actor and the critics are endowed with a four-layer fully-connected neural network (FCNN), respectively. Some tricks are used in both networks to improve the effectiveness of the algorithm. For the actor network, the input (i.e., individual states of each agent) should be normalized to avoid overfitting. In the output layer, decision elements should also be addressed via discretization or normalization methods, to satisfy the mixed decision variable constraints. For the critic networks, the input (i.e., action-state pairs) should also be normalized to contribute to better Q-value estimation.

Note that the target critics have the same structures as the evaluation critics.

3) The off-policy formulation is adopted to improve the sample efficiency, which is based on an experience replay technique. Specifically, an experience replay pool $\mathcal{K}_r$ endowed with a certain memory capacity is deployed. In the sampling stage, the experiences of all the agents are stored into the $\mathcal{K}_r$. In the training stage, the agents randomly sample a batch of experiences from $\mathcal{K}_r$ to train the network parameters in an off-policy manner, such that the sampled experiences can be effectively utilized to achieve convergence.

Multi-Agent RL Framework: According to the left part of Fig. 2, the edge agents operate at the corresponding edge controllers, respectively. Each agent can only observe the local information from its individual environment, and the decisions made by the agent are insensible to the others.

We adopt the mechanism of centralized training while decentralized execution [36], as illustrated in the central part of Fig. 2. The mechanism allows the policies to utilize additional information to simplify training process, on condition that this information is not utilized at execution stage. In the centralized training stage, except for the local information, additional information, including the states and actions of all the agents, is also available to each other. Specifically, the actor network of the edge agent captures the environment dynamics only from its own observed local state information, and then makes decisions for the individual environment. The critic network requires the action-state pairs of all the edge agents to generate the Q-values for evaluating the decisions. In the decentralized execution stage, the network parameters of the actor and the critics no longer need to be updated. Thus, only the local state information is required for the actor of each edge agent, the JDSRA decisions then can be obtained independently without being aware of other agents' state information.

Remark: Once well trained, the RL algorithm is deployed to make online decisions for the three-layer FL architecture, which is an inference process. The energy consumption of inference process of the proposed RL algorithm is very low since the adopted neural network has shallow layers, we therefore neglect the energy consumption consumed by the RL algorithm.

B. MASAC-Based Algorithm

In the following, we illustrate the learning procedure of the MASAC in detail, which is presented in Algorithm 2.

1) Initialization (Line 1): At the beginning of the centralized training stage, the parameters of the actor and critic networks of the edge agents are initialized, which will be updated in terms of the learning step. In addition, an experience replay pool $\mathcal{K}_r$ is instantiated.

2) Experience Sampling (Lines 6-9): The experience of the edge agents is denoted by a multi-dimensional tuple of the selected action, state transition, and feedback reward, i.e., $\{\boldsymbol{s}_{m,t}, \boldsymbol{a}_{m,t}, \boldsymbol{r}_{m,t}, \boldsymbol{s}'_{m,t}\}_{\forall m\in\mathcal{M}}$, which is obtained via the following steps. First, the edge agents observe the local state information $\{\boldsymbol{s}_{m,t}\}_{\forall m\in\mathcal{M}}$ from their individual environments, respectively. Second, the actors of the agents independently make the decisions $\{\boldsymbol{a}_{m,t}\}_{\forall m\in\mathcal{M}}$ with their own policy $\{\boldsymbol{\pi}_{\boldsymbol{\theta}_m}\}_{\forall m\in\mathcal{M}}$ according to the local information. Here, $\boldsymbol{a}_{m,t} = \{\boldsymbol{o}_{m,t}, \boldsymbol{\xi}^{\downarrow}_{m,t}, \boldsymbol{\xi}^{\uparrow}_{m,t}, \boldsymbol{\eta}_{m,t}\}$ represents the JDSRA decisions made by edge agent m for optimizing FL under the correlated BS. Then, the agents obtain the reward $\{\boldsymbol{r}_{m,t}\}_{\forall m\in\mathcal{M}}$, and individual environments evolve to the next state $\{\boldsymbol{s}'_{m,t}\}_{\forall m\in\mathcal{M}}$. Finally, the formulated transition tuple $\{\boldsymbol{s}_{m,t}, \boldsymbol{a}_{m,t}, \boldsymbol{r}_{m,t}, \boldsymbol{s}'_{m,t}\}_{\forall m\in\mathcal{M}}$ is stored in the $\mathcal{K}_r$ for algorithm parameter updating.

Algorithm 2 MASAC-Based Algorithm for the JDSRA

```
   Input: Remaining energy and time of all the IGWs
          $\{T^r_{m,n,t}, E^r_{m,n,t}\}_{\forall m\in\mathcal{M}, n\in\mathcal{N}_m}$;
   Output: JDSRA decision $\boldsymbol{a}_t = \{\boldsymbol{o}, \boldsymbol{\xi}^{\downarrow}, \boldsymbol{\xi}^{\uparrow}, \boldsymbol{\eta}\}$;
1  Initialize actor, evaluation critic, and target critic
   networks with weights $\boldsymbol{\theta}_m$, $\boldsymbol{\phi}_{m,1}$, $\boldsymbol{\phi}_{m,2}$, $\boldsymbol{\phi}'_{m,1}$, and $\boldsymbol{\phi}'_{m,2}$
   for each agent $m \in \mathcal{M}$, and experience replay pool $\mathcal{K}_r$;
2  for each episode do
3      Receive initial state $\boldsymbol{s}_1 = \{\boldsymbol{s}_{m,1}\}_{\forall m\in\mathcal{M}}$;
4      for FL epoch $t \in \mathcal{R}$ do
5          for each agent do
6              ▷ Experience sampling
7              Obtain decision $\boldsymbol{a}_{m,t}$ with policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_m}(\boldsymbol{s}_{m,t})$;
8              Execute decision $\boldsymbol{a}_{m,t}$, and obtain reward $\boldsymbol{r}_{m,t}$
               by Algorithm 1 and next state $\boldsymbol{s}'_{m,t}$;
9              Store $\{\boldsymbol{s}_{m,t}, \boldsymbol{a}_{m,t}, \boldsymbol{r}_{m,t}, \boldsymbol{s}'_{m,t}\}$ into $\mathcal{K}_r$ and
               replace the oldest experiences in $\mathcal{K}_r$;
10             ▷ Parameter updating
11             Randomly sample a minibatch of $K_b$
               experiences $\{\boldsymbol{s}^k_{m,t}, \boldsymbol{a}^k_{m,t}, \boldsymbol{r}^k_{m,t}, \boldsymbol{s}'^k_{m,t}\}$ from $\mathcal{K}_r$;
12             Set the target Q-value $y^k_{m,t}$ by Eq. (24);
13             Update $\boldsymbol{\phi}_{m,1}$ and $\boldsymbol{\phi}_{m,2}$ by minimizing the loss
               function in Eq. (23);
14             Update $\boldsymbol{\theta}_m$ via the policy gradient in Eq. (28);
15             Adjust $\alpha_m$ via the gradient obtained by
               Eq. (29);
16         Update target networks for the agents by Eq. (30);
17         Obtain cooperative decision $\boldsymbol{a}_t = \{\boldsymbol{a}_{m,t}\}_{\forall m\in\mathcal{M}}$.
```

3) FCNN Parameter Updating (Lines 10-17): In this stage, the edge agents are coordinated, and the algorithm alternates between collecting experience from all the individual environments with the current agents' policies and updating the FCNNs based on the batch of K_b transitions $\{\boldsymbol{s}^k_{m,t}, \boldsymbol{a}^k_{m,t}, \boldsymbol{r}^k_{m,t}, \boldsymbol{s}'^k_{m,t}\}_{\forall m\in\mathcal{M}}$ sampled from the $\mathcal{K}_r$. For edge agent m, the detailed update procedure is given as follows.

Firstly, the parameters $\boldsymbol{\phi}_{m,v}, \forall v \in \{1,2\}$ of the evaluation critic networks are independently updated via minimizing the loss function $\mathcal{L}(\boldsymbol{\phi}_{m,v})$, which is given by

$$\mathcal{L}(\boldsymbol{\phi}_{m,v}) = \frac{1}{2K_b}\sum_{k=1}^{K_b}\Big(\min_{v=1,2} Q_{\boldsymbol{\phi}_{m,v}}(\boldsymbol{s}^k_t, \boldsymbol{a}^k_{1,t}, \cdots, \boldsymbol{a}^k_{M,t}) - y^k_{m,t}\Big)^2, \tag{23}$$

where

$$y^k_{m,t} = r^k_{m,t} + \gamma(\min_{v=1,2} Q_{\phi'_{m,v}}(\boldsymbol{s}'^k_t, \boldsymbol{a}'^k_{1,t}, \cdots, \boldsymbol{a}'^k_{M,t}) - \alpha_m \log(\boldsymbol{\pi}_{\boldsymbol{\theta}_m}(\boldsymbol{a}'^k_{1,t}, \cdots, \boldsymbol{a}'^k_{M,t}|\boldsymbol{s}'^k_t))) \quad (24)$$

denotes the target Q-value. Here, $\boldsymbol{s}^k_t = \{\boldsymbol{s}^k_{m,t}\}_{\forall m \in \mathcal{M}}$ and $\boldsymbol{s}'^k_t = \{\boldsymbol{s}'^k_{m,t}\}_{\forall m \in \mathcal{M}}$ are the current and the next state sets of all the agents, $\log(\cdot)$ is the function to return the actions' entropy value, and $Q_{\phi_{m,v}}(\cdot)$ and $Q_{\phi'_{m,v}}(\cdot)$ are the state-action Q-values calculated by the evaluation and target critic networks. Note that two critics with parameters $\boldsymbol{\phi}_{m,v}$ and $\boldsymbol{\phi}'_{m,v}, \forall v \in \{1,2\}$ are deployed in both the evaluation and target networks, respectively, to alleviate positive bias in policy improvement. More importantly, only the minimum of these two Q-values is used for the loss calculation. As such, the stochastic gradient can be obtained for updating the parameter of the critic networks, i.e.,

$$\begin{aligned}\nabla_{\boldsymbol{\phi}_{m,v}} \mathcal{L}(\boldsymbol{\phi}_m) = &\nabla_{\boldsymbol{\phi}_{m,v}} Q_{\boldsymbol{\phi}_{m,v}}(\boldsymbol{s}^k_t, \boldsymbol{a}^k_{1,t}, \cdots, \boldsymbol{a}^k_{M,t}) \\ &\cdot(Q_{\boldsymbol{\phi}_{m,v}}(\boldsymbol{s}^k_t, \boldsymbol{a}^k_{1,t}, \cdots, \boldsymbol{a}^k_{M,t}) \\ &-(r^k_{m,t} + \gamma(Q_{\boldsymbol{\phi}'_{m,v}}(\boldsymbol{s}'^k_t, \boldsymbol{a}'^k_{1,t}, \cdots, \boldsymbol{a}'^k_{M,t}) \\ &-\alpha_m \log(\boldsymbol{\pi}_{\boldsymbol{\theta}_m}(\boldsymbol{a}'^k_{1,t}, \cdots, \boldsymbol{a}'^k_{M,t}|\boldsymbol{s}'^k_t))))). \end{aligned} \quad (25)$$

Secondly, the actor network is updated via policy gradient methods, and the objective function is defined as

$$J(\boldsymbol{\theta}_m) = -\mathbb{E}_{\boldsymbol{s}' \sim \mathcal{K}_b}[\mathbb{E}_{\boldsymbol{a}' \sim \boldsymbol{\pi}_{\boldsymbol{\theta}_m}}[\min_{v=1,2} Q_{\boldsymbol{\phi}'_{m,v}}(\boldsymbol{s}'^k_t, \boldsymbol{a}'^k_{1,t}, \cdots, \boldsymbol{a}'^k_{M,t}) -\alpha_m \log(\boldsymbol{\pi}_{\boldsymbol{\theta}_m}(\boldsymbol{a}'^k_{1,t}, \cdots, \boldsymbol{a}'^k_{M,t}|\boldsymbol{s}'^k_t))]]. \quad (26)$$

Here, the policy is reparameterized via an FCNN denoted by $g_{\boldsymbol{\theta}_m}(\epsilon_t; \boldsymbol{s}_{m,t})$, in which an input noise ϵ_t is added to obtain a lower variance estimation. Hence, Eq. (26) can be rewritten as

$$J(\boldsymbol{\theta}_m) = -\mathbb{E}_{\boldsymbol{s}' \sim \mathcal{K}_b, \boldsymbol{\epsilon}_t \sim \boldsymbol{\pi}_{\boldsymbol{\theta}_m}}[\min_{v=1,2} Q_{\boldsymbol{\phi}'_{m,v}}(\boldsymbol{s}'^k_t, g_{\boldsymbol{\theta}_m}(\epsilon_t; \boldsymbol{s}^k_t)) -\alpha_m \log(\boldsymbol{\pi}_{\boldsymbol{\theta}_m}(g_{\boldsymbol{\theta}_m}(\epsilon_t; \boldsymbol{s}^k_t)|\boldsymbol{s}'^k_t))], \quad (27)$$

where ϵ_t is sampled from a Gaussian distribution. Thus, the gradient of the policy can be calculated, i.e.,

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_m} J(\boldsymbol{\theta}_m) = &\nabla_{\boldsymbol{\theta}_m} \alpha_m \log(\boldsymbol{\pi}_{\boldsymbol{\theta}_m}(\boldsymbol{a}^k_{1,t}, \cdots, \boldsymbol{a}^k_{M,t}|\boldsymbol{s}^k_t)) \\ &+(\nabla_{\boldsymbol{a}_{m,t}} \alpha_m \log(\boldsymbol{\pi}_{\boldsymbol{\theta}_m}(\boldsymbol{a}^k_{1,t}, \cdots, \boldsymbol{a}^k_{M,t}|\boldsymbol{s}^k_t)) \\ &-\nabla_{\boldsymbol{a}_{m,t}} Q(\boldsymbol{s}^k_t, \boldsymbol{a}^k_{1,t}, \cdots, \boldsymbol{a}^k_{M,t}))\nabla_{\boldsymbol{\theta}_m} g_{\boldsymbol{\theta}_m}(\epsilon_t; \boldsymbol{s}^k_t). \end{aligned} \quad (28)$$

Thirdly, instead of choosing the weight α_m manually, an FCNN is adopted to automate the weight set for the maximum entropy objective. The gradients of α_m can be computed by the following objective, i.e.,

$$J(\alpha_m) = \mathbb{E}_{\boldsymbol{a}_t \sim \boldsymbol{\pi}_{\boldsymbol{\theta}_m}}\left[-\alpha_m \log(\boldsymbol{\pi}_{\boldsymbol{\theta}_m}(\boldsymbol{a}^k_t|\boldsymbol{s}^k_t)) - \alpha_m H'\right], \quad (29)$$

where $\boldsymbol{a}^k_t = \{\boldsymbol{a}^k_{m,t}\}_{\forall m \in \mathcal{M}}$ is the decision set of all the agents, and H' is the value of target entropy.

Finally, to stabilize the learning process, the parameters $\boldsymbol{\phi}'_{m,v}$ of target critic networks are updated from the evaluation critics' parameters $\boldsymbol{\phi}_{m,v}$ via a soft-updating method, i.e.,

$$\boldsymbol{\phi}'_{m,v} = \tau \boldsymbol{\phi}_{m,v} + (1-\tau)\boldsymbol{\phi}'_{m,v}, \quad \forall v \in \{1,2\}, \quad (30)$$

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

Algorithm 3 Device Refinement Subroutine

```
**Input**: Decision $\{\boldsymbol{o}, \boldsymbol{\xi}^{\downarrow}, \boldsymbol{\xi}^{\uparrow}, \boldsymbol{\eta}\}$ obtained by **Algorithm 2**;
**Output**: Refined participating device set $\mathcal{N}^\star_m$;
1 Select an initial set of $\mathcal{N}_m, \forall m \in \mathcal{M}$ IGWs for each
  edge server with device selection decision $\boldsymbol{o}$;
2 Allocate downlink $\boldsymbol{\xi}^{\downarrow}$ and uplink $\boldsymbol{\xi}^{\uparrow}$ spectrum and
  on-device computing $\boldsymbol{\eta}$ resources for the selected IGWs;
3 Calculate the remaining energy $E^r_{m,n,t}$ and time budget
  $T^r_{m,n,t}$, and FL epoch delay $T_t$;
4 Set $\mathcal{N}^\star_m = \varnothing$;
5 for each $BS\ m \in \mathcal{M}$ do
6     for each $IGW\ n \in \mathcal{N}_m$ do
7         if $E^r_{m,n,t} \le E^{\max}_{m,n}$, $T^r_{m,n,t} \le T^{\max}_{m,n}$, and $T_t \le T^{\max}$
          then
8             $\mathcal{N}^\star_m \leftarrow \mathcal{N}^\star_m \cup \{(m,n)\}$.
```

C. Device Refinement Subroutine

In each FL epoch, Algorithm 2 generates the JDSRA decisions to support the FL operation. Based on these decisions, a refinement subroutine is adopted to obtain a refined participating device set that can complete the FL with satisfied energy and delay requirements, thereby accelerating FL convergence. The subroutine is operated as the following steps.

1) Remaining On-Device Resource Calculation (Lines 1-4): After obtaining the JDSRA decisions, the edge controllers can select an initial set of $\mathcal{N}_m, \forall m \in \mathcal{M}$ IGWs to participate in the FL with the allocated spectrum and computing resources. And then, the remaining on-device energy and IGWs' time budget can be calculated, which are consumed by the FL parameter transmission and local model training.

2) Participating Device Set Refinement (Lines 5-8): With the above calculations, the edge controllers can evaluate whether the selected IGWs are able to complete the FL while satisfying the strict requirements. Only when the IGWs can satisfy the above requirement, the system would allow these IGWs to participate in the FL. Thus, a refined set of participating IGWs $\mathcal{N}^\star_m, \forall m \in \mathcal{M}$ can be obtained to optimize FL.

With this subroutine, the on-device energy consumption and time budget can be effectively reduced to accelerate convergence in the distributed and resource-limited IIoT networks.

D. Computational Complexity Analysis

As described before, the MASAC algorithm contains M edge agents. Each agent is endowed with one actor network, four critic networks, and one α_m weight networks. Each of them is represented by an FCNN, as shown in Fig. 3. Generally, the computational complexity of an FCNN is $K = \mathcal{O}(\sum_x (2I_x - 1)I_{x+1})$ [37], where $x \in [0, L]$ denotes the layer index and I_x represents the neuron number of hidden layers. According to the different input-output structures, there are three cases for the above networks. Specifically,

1) For actor networks, $I_1 = 2|\mathcal{N}_m|, \forall m \in \mathcal{M}$ is the dimension of the corresponding individual environment states, and $I_L = 4|\mathcal{N}_m|$ is the decision dimension of edge agent m;

2) For critic networks, $I_1 = (2+4)\sum_m |\mathcal{N}_m|$ is the dimension of total states and decisions. Here, the total states are composed of all the individual states observed by the agents of dimension $2\sum_m |\mathcal{N}_m|$. The decisions include device selection variable, and uplink/downlink spectrum and computing resource allocation variables with dimension $4\sum_m |\mathcal{N}_m|$. Note that the output of the critic network is an estimated Q-value with dimension 1, and thus $I_L = 1$;

3) For α_m weight networks, the network scale is very small, and thus the computational complexity can be neglected.

In summary, there are $5 \times M$ FCNNs should be considered in the MASAC, thus the total computational complexity of the algorithm is $5 \times M \times K$.

Moreover, the proposed RoF scheme can reduce signaling overhead. During the FL operation, the central controller deployed at the cloud server collects the network state information from IGWs in each epoch. In the single-agent RoF scheme, the state information has to be sent to the central controller. In the RoF scheme, the state information only needs to be sent to the edge controllers that are deployed at edge servers. That is, compared with the single-agent RoF scheme, the RoF scheme can effectively reduce the signaling overhead of reporting all the state information to the cloud server.

VII. Performance Evaluation

In this section, simulations are conducted to evaluate the proposed RoF scheme for optimizing FL in the distributed IIoT networks.

A. Experimental Setup

We consider three smart factories operating in different cities, respectively, and 30 IGWs are distributed in each factory that is covered by a BS unless specified. The channel gain in IIoT networks is modeled by $PL(\text{dB}) = -128.1 - 37.6\log_{10}(l)$, where l denotes the transmission distance between the BS and the covered IGWs [38]. We adopt Raspberry Pi 4B as an IGW [39]. The effective capacitance is set to be 2×10^{-28}, and the number of CPU cycles to execute one training sample is set to be 20 cycles/bit [28]. For the MASAC algorithm, the actor and the critics of each edge agent are with [128, 256, 128, 120] and [1024, 512, 256, 1] neurons, respectively. Note that each agent employs the same actor and critic network architecture. Other important simulation parameters are listed in Table I.

For the FL, we design a convolutional neural network (CNN)-based model [41], which includes two 5×5 convolutional layers, two fully-connected layers, and a softmax output layer (61,366 total trainable model parameters), to perform classification tasks (i.e., fault diagnostic). In addition, the learning rate and batch size of the CNN model are set to be 2.5×10^{-3} and 64, respectively. The regularization coefficient is set to be 5×10^{-5}. In this simulation, the CNN is trained to perform fault diagnostic task, and thus the loss function can be expressed as

$$f(\boldsymbol{x}_d, y_d; \boldsymbol{w}) = \sum_{z=0}^{Z} p(y_d = z)\mathbb{E}_{\boldsymbol{x}|y_d=z}[\log \hat{y_d}], \tag{31}$$

TABLE I

Simulation Parameters [35], [40]

Network Para.	**Value**	**Learning Para.**	**Value**
$p_{m,n}$, p_m	30, 40 dBm	*Optimizer*	Adam
σ^2	-104 dBm	*Episode number*	300
l	150 m	*Epoch number*	300
$B^{\uparrow}$, $B^{\downarrow}$	20, 40 MHz	K_r	10,000
f	1.5 GHz	K_b	64
R^c	1 Mbps	γ	0.9
$E_{m,n}^{\max}$	[4500, 5000] Joule	ϵ	3×10^{-4}
$T_{m,n}^{\max}$	[600, 800] s	τ	0.01
$T^{\max}$	5 s	U	10

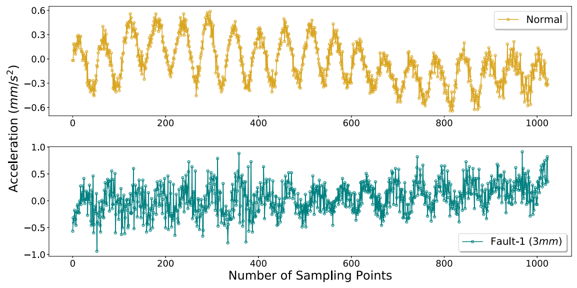


Fig. 4. Two classes of samples in AWE dataset. These two samples are collected from rolling bearings of industrial facilities. “Normal” means the bearing stays in a health condition, and “Fault-1” means the bearing operates under a fault condition with 3 mm severity degree.

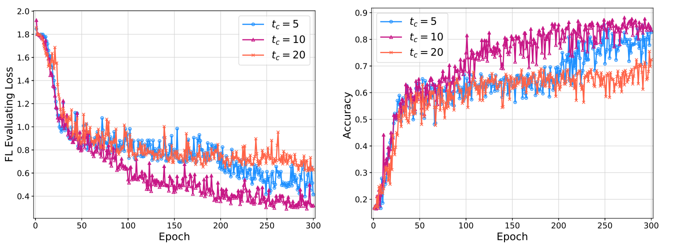
where Z is the number of health condition classes, y_d is the true label, and $\hat{y_d}$ is the predicted label [21].

The data size of model parameters S_g is 240 KB. In addition, the CNN model is trained on the AWE dataset,[2] which is a vibration-based industrial sensor dataset contains six health condition classes of industrial facilities [42]. In our simulations, the number of data samples in the AWE dataset is 23,436, and the ratio between the training set and evaluating set is 9:1. Each sample contains 1,024 sampling points. Fig. 4 shows two raw samples of the AWE dataset, which are collected from the “Normal” and “Fault-1” health conditions, respectively. In the considered scenario, FL is operated in a non-IID setting, in which each IGW only holds a few classes of samples. To achieve this, we sort the samples by the class label, partition them into 180 blocks with the same size, and assign 2 blocks to each IGW [10].

To evaluate the performance of the proposed scheme, the following benchmarks are compared.

- *Single-Agent RoF (SARoF)*: In this scheme, we assume a central controller is deployed at the cloud server and acts as the agent to centrally select participating IGWs and manage the spectrum and computing resources for the FL. In the SARoF, the hidden layers of the actor network are with [256, 512, 384, 360] neurons, and the critic networks have the same structure as that of the RoF.
- *The RoF Without Edge Aggregation (RoF w.o.)*: This scheme is designed for a two-layer FL aggregation

[2]Online available. https://github.com/Intelligent-AWE/DeepHealth



(a) Loss (b) Accuracy

Fig. 5. FL evaluating loss and accuracy with respect to different cloud aggregation frequencies.

architecture, in which the FL model parameters are only aggregated by the cloud server in each FL epoch.

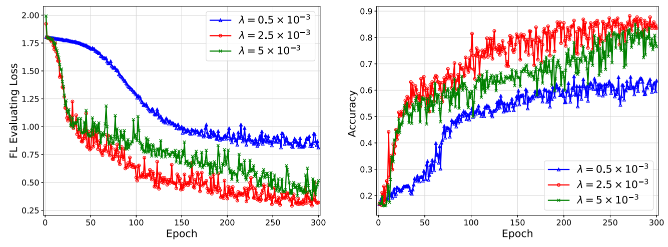
- *Multi-Agent DDPG (MADDPG)*: In this scheme, the hidden layers of the actor networks and the critic networks are with [180, 300] and [450, 100] neurons, respectively. Multiple learning agents are deployed at the edge servers to cooperatively make the JDSRA decisions via a DDPG algorithm.
- *Random Probabilistic Configuration (RPC)*: In this scheme, the random policy is adopted, in which the central controller randomly selects the IGWs and allocates the spectrum and computing resources. All available actions are selected with an equal probability.

B. Evaluation of RoF Scheme

1) Convergence Performance: We evaluate the convergence performance of the three-layer collaborative FL for the CNN-based fault diagnostic model. During the FL operation, the cloud aggregation is performed every t_c FL epochs. As shown in Fig. 5, the CNN model trained on the AWE dataset can achieve a low evaluating loss and a high accuracy over the considered three-layer collaborative architecture within 300 FL epochs. In particular, lower FL evaluating loss and higher accuracy can be reached if the cloud aggregation frequency is set properly. It can be seen that the lowest evaluating loss and the highest accuracy are respectively obtained when the cloud aggregation frequency t_c is set to be 10. Moreover, due to the non-IID setting and the limited on-device energy, the FL may not benefit from a large cloud aggregation frequency. This is because on-device energy might be exhausted before cloud aggregation, such that the model parameters trained on these IGWs cannot be globally aggregated at the cloud server, which slows down global model convergence.

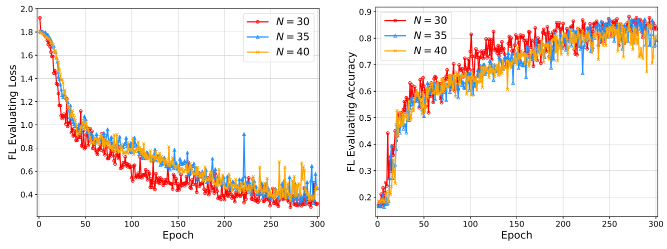
As shown in Fig. 6, the convergence performance of the CNN-based model in terms of learning rates is evaluated. It can be seen that the lowest evaluating loss and the highest accuracy are respectively achieved when the learning rate is set to be 2.5×10^{-3}. The result indicates that a larger learning rate may not contribute to achieve a better performance. Also, a smaller value of learning rate can lead to a stable FL process but is not good for achieving a lower loss, and thus may consume more spectrum and computing resources to achieve convergence.

To demonstrate the scalability of the proposed solution, we evaluates the performance of the RoF scheme in terms



(a) Loss (b) Accuracy

Fig. 6. FL evaluating loss and accuracy with respect to different learning rates.

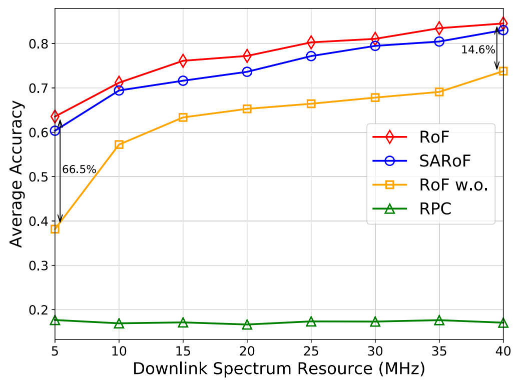


(a) Loss (b) Accuracy

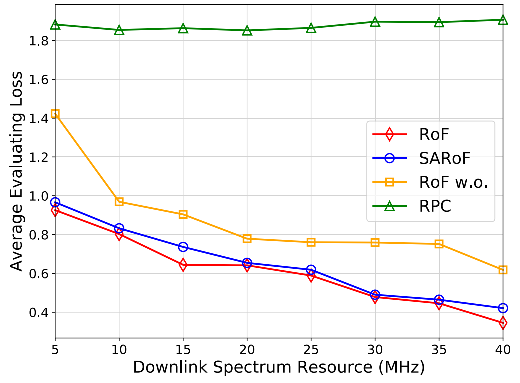
Fig. 7. FL evaluating loss and accuracy with respect to different device (i.e., the IGW) number.

of different numbers of IGWs dispersed in each factory. As shown in Fig. 7, the CNN model can achieve a low evaluating loss and a high accuracy within 300 FL epochs. In particular, the achieved evaluating loss and accuracy do not degrade much as the increase of the number of IGWs. The result indicates that the presented RoF scheme can be extended to dense networks with a large number of IGWs. However, with the increasing of the number of IGWs, the decision dimension of the RoF scheme increases, which increases the difficulty for the optimal decision making.

2) Impact of Downlink Spectrum Resources: To demonstrate the effectiveness of the learning-based RoF scheme, we evaluate the average accuracy and evaluating loss of different schemes with respect to different amounts of downlink spectrum resources, as shown in Fig. 8. As expected, the RoF achieves the highest average accuracy and lowest average evaluating loss among all the schemes. Specifically, the accuracies incurred by the learning-based schemes (i.e., RoF, SARoF, and RoF w.o.) increase with the growth of the amounts of downlink spectrum resources. On the contrary, the evaluating loss achieved by the learning-based schemes decreases with the decrease of the amounts of downlink spectrum resources. The accuracy achieved by the RoF is approximately 1.8% higher than that by the SARoF scheme. The result indicates that the RoF can achieve a higher resource utilization and is more scalable compared with other schemes. In addition, the RoF can increase the average accuracy by 14.6%, as compared with the RoF without edge aggregation. This is because two-layer (i.e., device-cloud) FL architecture leads to higher parameter transmission delay, which prevents the cloud server from aggregating more model parameters with the FL strict epoch delay constraint and thus against FL performance improvement.



(a) Accuracy



(b) Loss

Fig. 8. Average accuracy and evaluating loss comparison among different schemes with respect to downlink spectrum resource.

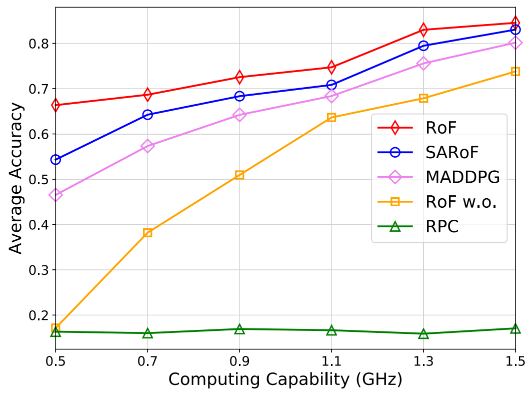


Fig. 9. Average accuracy comparison among different schemes with respect to computing capability.

3) Impact of Computing Capabilities: As shown in Fig. 9, the average accuracy of different algorithms in terms of on-device computing capabilities is evaluated. The result shows that all learning-based schemes, i.e., RoF, SARoF, MADDPG, and RoF w.o., can achieve higher accuracy than that of RPC benchmark. This is because that learning-based schemes make the FL models converge to a high accuracy level via judicious computing resource allocation. In addition, with the increase of the available computing capabilities, the average accuracies achieved by the RoF, SARoF, MADDPG, and

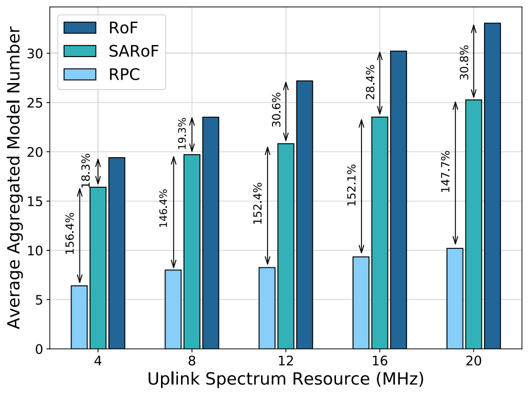


Fig. 10. Average aggregated model number comparison among different schemes with respect to uplink spectrum resource.

RoF w.o. schemes increase. The underlying reason is that the delay of local model training can be greatly reduced with more available computing resources, thereby ensuring the FL can be completed within strict delay requirements. Particularly, the proposed multi-agent empowered RoF scheme can achieve the highest average accuracy than the benchmarks. When the amount of computing capability is 1.5 GHz, the RoF can increase the average accuracy by 1.81%, 5.47%, and 14.56% compared with the SARoF, MADDPG, and RoF w.o. benchmarks, respectively.

4) Impact of Uplink Spectrum Resources: Efficient uplink spectrum resource utilization ensures the parameter aggregation can be completed within the strict FL epoch delay requirement. As shown in Fig. 10, the impact of uplink spectrum resources on the aggregated model number of different schemes is evaluated. Intuitively, more IGWs' model parameters can be aggregated by the edge or cloud servers when there are more available uplink spectrum resources. More importantly, the RoF achieves the maximum average aggregated model number among all the schemes, implying that more aggregated model parameters can be utilized to the optimal global FL model. In particular, when the amount of uplink spectrum resource is 20 MHz, the RoF increases the average number of aggregated model by 30.8% compared with the SARoF scheme. This is because the RoF can better allocate the uplink spectrum resources, thereby enabling the parameter uploading to be completed with an acceptable transmission delay. With more model parameters aggregated from the distributed IGWs, the FL models can achieve a higher accuracy level, as the red curve shown in Fig. 8. The result indicates that implementing FL over more usable data can effectively optimize the FL performance.

5) Energy Consumption and Time Analysis: As shown in Fig. 11, the average energy consumption and time of different algorithms in terms of uplink spectrum resource are presented. The RoF can effectively reduce the average energy consumption and time as compared to the SARoF scheme. This is because the proposed RoF scheme can better utilize uplink spectrum resources to enable FL models to be converged within energy consumption and time requirements in a decentralized resource management manner. In addition, both

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT SCHEMES

	Average Accuracy	Average Loss	Average Participating Num.	Average Aggregated Num.	Average Energy (Joule)	Average Time (s)
RoF	84.5%	0.3451	53.8	33.1	2550.3	413.8
SARoF	83.1%	0.4209	36.5	25.3	2967.2	433.5
RoF w.o.	73.8%	0.62	36.2	10.9	2539.3	190.5
RPC	17.0%	1.81	44.8	10.2	4800	11.5

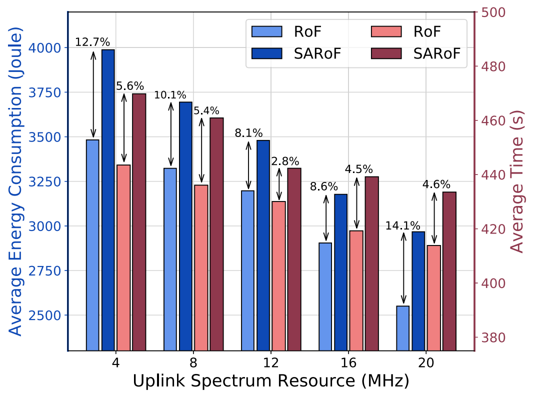
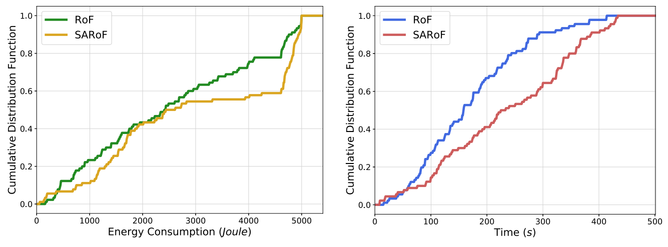


Fig. 11. Average energy consumption and time of different schemes with respect to uplink spectrum resource.



(a) Energy (b) Time

Fig. 12. Cumulative distribution functions of energy consumption and time during the FL process.

the energy consumption and time decrease with the increase of the amounts of available spectrum resource. Specifically, when the amount of uplink spectrum resource is 20 MHz, the RoF reduces the average energy consumption and time within the FL process by approximately 14.1% and 4.6%, respectively. The reason is that sufficient spectrum resource enables the model parameters to be aggregated at the edge or cloud with a shorter parameter uploading time, thereby consuming less on-device energy for parameter uploading. In Fig. 12, the cumulative distribution functions of the energy consumption and time of different schemes are compared, which shows that the RoF scheme can better guarantee the energy consumption and time requirements.

6) Comprehensive Performance Analysis: As listed in Table II, a comprehensive comparison is presented to demonstrate the performance of different schemes. Overall, three important observations can be obtained from the simulation results. Firstly, the RoF achieves the optimal performance over all the metrics given multiple constrained resources. Specifically, the average accuracy of the RoF is about 84.5%, which is higher than that of the SARoF (83.1%), RoF w.o. (73.8%), and RPC (17.0%). The average loss of the RoF is about 0.35, which is lower than that of the SARoF (0.42), RoF w.o. (0.62), and RPC (1.81). Secondly, aiming at achieving fast FL convergence, the RoF is inclined to select more IGWs to participate in the FL. However, it is interesting to note that, the ratio between the number of aggregated model and the number of participating IGW for RoF is 61.4%, which is lower than that of SARoF (69.3%). Constrained computing and spectrum resources have significant effects on the local model training and parameter transmission, which should be better utilized to support the long-term FL process, thereby providing reliable distributed learning service. Thirdly, although more IGWs are selected by the RoF during the FL process, they consumes less energy and time than the SARoF. The result implies that the RoF can allocate the cherished resources in a more judicious way, which endows more persistence to the system. The above simulation results indicate that the considered trade-offs should be better balanced to facilitate efficient FL in the distributed IIoT networks.

Based on the above observations, we can claim that the proposed RoF scheme can effectively improve the FL performance in the distributed and resource-limited IIoT networks.

VIII. CONCLUSION

In this paper, we have investigated the FL performance optimization problem in the distributed IIoT networks. To solve the problem, we have proposed the RoF scheme, based on deep multi-agent reinforcement learning, to make the optimal device selection and resource allocation decisions in an online manner. Simulation results based on real-world dataset demonstrate that the proposed scheme can achieve a high FL accuracy, while satisfying on-device energy consumption requirements. The RoF scheme operates in a distributed manner, which can effectively reduce system signaling overhead of collecting network information, especially in resource-constrained networks. For the future work, we will study a data importance-aware device selection scheme to optimize FL in a large-scale IIoT network.

REFERENCES

[1] W. Zhang, D. Yang, W. Wu, H. Peng, H. Zhang, and X. S. Shen, “Spectrum and computing resource management for federated learning in distributed industrial IoT,” in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Montreal, QC, Canada, Jun. 2021, pp. 1–6.
[2] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, “Artificial neural networks-based machine learning for wireless networks: A tutorial,” *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3039–3071, 4th Quart., 2019.

[3] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[4] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, 2019.

[5] X. S. Shen *et al.*, "Data management for future wireless networks: Architecture, privacy preservation, and regulation," *IEEE Netw.*, vol. 35, no. 1, pp. 8–15, Jan. 2021.

[6] D. Gunduz, P. de Kerret, N. Sidiroupoulos, D. Gesbert, C. Murthy, and M. van der Schaar, "Machine learning in the air," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2184–2199, Oct. 2019.

[7] K. Huang, G. Zhu, C. You, J. Zhang, Y. Du, and D. Liu, "Communication, computing, and learning on the edge," in *Proc. IEEE Int. Conf. Commun. Syst. (ICCS)*, Chengdu, China, Oct. 2018, pp. 268–273.

[8] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.

[9] M. Bennis, "Federated learning and control at the wireless network edge," *GetMobile, Mobile Comput. Commun.*, vol. 24, no. 3, pp. 9–13, Jan. 2021.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. PMLR*, Tallahassee, FL, USA, 2017, pp. 1273–1282.

[11] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time minimization of federated learning over wireless networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–6.

[12] M. M. Wadu, S. Samarakoon, and M. Bennis, "Joint client scheduling and resource allocation under channel uncertainty in federated learning," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5962–5974, Sep. 2021, doi: 10.1109/TCOMM.2021.3088528.

[13] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 1205–1221, Jun. 2019.

[14] 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.

[15] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 72–80, Apr. 2020.

[16] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[17] X. Chen *et al.*, "Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2377–2392, Oct. 2019.

[18] L. Chen and J. Xu, "Seek common while shelving differences: Orchestrating deep neural networks for edge service provisioning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 251–264, Jan. 2021.

[19] H. Peng and X. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2416–2428, Oct. 2020.

[20] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.

[21] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun.*, Beijing, China, Jul. 2020, pp. 1698–1707.

[22] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient radio resource allocation for federated edge learning," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Dublin, Ireland, Jun. 2020, pp. 1–6.

[23] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2282–2292, Oct. 2019.

[24] 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.

[25] N. Zhao, H. Wu, F. R. Yu, L. Wang, W. Zhang, and V. C. M. Leung, "Deep reinforcement learning-based latency minimization in edge intelligence over vehicular networks," *IEEE Internet Things J.*, early access, May 10, 2021, doi: 10.1109/JIOT.2021.3078480.

[26] K. Li, W. Ni, E. Tovar, and A. Jamalipour, "On-board deep Q-network for UAV-assisted online power transfer and data collection," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 12215–12226, Dec. 2019.

[27] A. Kaur and K. Kumar, "Energy-efficient resource allocation in cognitive radio networks under cooperative multi-agent model-free reinforcement learning schemes," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 3, pp. 1337–1348, Sep. 2020.

[28] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*. Dublin, Ireland, Jun. 2020, pp. 1–6.

[29] W. Wu, N. Cheng, N. Zhang, P. Yang, W. Zhuang, and X. Shen, "Fast mmWave beam alignment via correlated bandit learning," *IEEE Trans. Wireless Commun.*, vol. 18, no. 12, pp. 5894–5908, Dec. 2019, doi: 10.1109/TWC.2019.2940454.

[30] F. Luo and C. Zhang, *Signal Processing for 5G: Algorithms and Implementations*. Hoboken, NJ, USA: Wiley, 2016.

[31] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*, Paris, France, 2010, pp. 177–186.

[32] S. Deng *et al.*, "Dynamical resource allocation in edge for trustable Internet-of-Things systems: A reinforcement learning method," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6103–6113, Sep. 2020.

[33] X. Shen *et al.*, "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 45–66, 2020.

[34] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 729–743, Feb. 2020.

[35] 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. ICML* Stockholm, Sweden, 2018, pp. 1861–1870.

[36] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. NeurIPS*, Long Beach, CA, USA, 2017, pp. 1–4.

[37] R. Livni, S. Shalev, and O. Shamir, "On the computational efficiency of training neural networks," in *Proc. NeurIPS*, vol. 27, Montréal, QC, Canada, 2014, pp. 855–863.

[38] G. Zhao, B. Huang, L. Li, and X. Zhou, "Relay-assisted cross-channel gain estimation for spectrum sharing," *IEEE Trans. Commun.*, vol. 64, no. 3, pp. 973–986, Mar. 2016.

[39] *Raspberry Pi 4 Model B*. Accessed: Jun. 21, 2019. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/rpi_DATA_2711_1p0_preliminary.pdf

[40] H. Peng and X. Shen, "Multi-agent reinforcement learning based resource management in MEC-and UAV-assisted vehicular networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 131–141, Jan. 2021.

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[42] W. Zhang, D. Yang, Y. Xu, X. Huang, J. Zhang, and M. Gidlund, "DeepHealth: A self-attention based method for instant intelligent predictive maintenance in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5461–5473, Aug. 2021.

Weiting Zhang (Student Member, IEEE) is currently pursuing the Ph.D. degree in communication and information systems with Beijing Jiaotong University, Beijing, China. From November 2019 to November 2020, he was a Visiting Ph.D. Student with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include mobile edge computing, the Industrial Internet of Things, and machine learning for wireless networks.

Dong Yang (Member, IEEE) received the B.S. degree from Central South University, Hunan, China, in 2003, and the Ph.D. degree in communications and information science from Beijing Jiaotong University, Beijing, China, in 2009. From March 2009 to June 2010, he was a Post-Doctoral Research Associate with Jönköping University, Jönköping, Sweden. In August 2010, he joined the School of Electronic and Information Engineering, Beijing Jiaotong University. Since 2017, he has been a Full Professor of communication engineering with Beijing Jiaotong University. His research interests include network architecture, wireless sensor networks, industrial networks, and the Internet of Things.

Wen Wu (Member, IEEE) received the B.E. degree in information engineering from the South China University of Technology, Guangzhou, China, in 2012, the M.E. degree in electrical engineering from the University of Science and Technology of China, Hefei, China, in 2015, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2019. Since 2019, he has been working as a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, University of Waterloo. His research interests include millimeter-wave networks and AI-empowered wireless networks.

Haixia Peng (Member, IEEE) received the Ph.D. degree in computer science and electrical and computer engineering from Northeastern University, China, in 2017, and the University of Waterloo, Canada, in 2021. She is currently an Assistant Professor with the Department of Computer Engineering and Computer Science, California State University Long Beach, Long Beach, CA, USA. She has authored or coauthored more than 30 technical papers dealing with network issues. Her current research focuses on the Internet of Vehicles, resource management, multi-access edge computing, and reinforcement learning. She serves/served as a TPC Member for IEEE ICC, Globecom, and VTC conferences, and a Reviewer for IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC), IEEE TRANSACTIONS ON COMMUNICATIONS, and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and more than 20 prestigious journals.

Ning Zhang (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2015. After that, he was a Post-Doctoral Research Fellow with the University of Waterloo and the University of Toronto, Canada. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Windsor, Canada. His research interests include connected vehicles, mobile edge computing, wireless networking, and machine learning. He is a Highly Cited Researcher. He received the NSERC PDF Award in 2015 and six Best Paper Awards from IEEE Globecom in 2014, IEEE WCSP in 2015, IEEE ICC in 2019, IEEE ICCC in 2019, the IEEE Technical Committee on Transmission Access and Optical Systems in 2019, and *Journal of Communications and Information Networks* in 2018. He also serves/served as the General Chair for IEEE SAGC 2021, the TPC Chair for IEEE VTC 2021 and IEEE SAGC 2020, the Track Chair for several international conferences, including IEEE ICC 2022, CollaborateCom 2021, IEEE VTC 2020, AICON 2020 and CollaborateCom 2020, and the Co-Chair for numerous international workshops. He serves as an Associate Editor for IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, and IEEE SYSTEMS JOURNAL, and a Guest Editor for several international journals, such as *IEEE Wireless Communications*, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, and IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING.

Hongke Zhang (Fellow, IEEE) received the M.S. and Ph.D. degrees in electrical and communication systems from the University of Electronic Science and Technology of China, Chengdu, China, in 1988 and 1992, respectively. From 1992 to 1994, he was a Post-Doctoral Researcher with Beijing Jiaotong University, Beijing, China, where he is currently a Professor with the School of Electronic and Information Engineering and the Director of the National Engineering Lab on Next Generation Internet Technologies. He has authored more than ten books and the holder of more than 70 patents. His research has resulted in many papers, books, patents, systems, and equipment in the areas of communications and computer networks. He is the Chief Scientist of the National Basic Research Program of China (973 Program) and has also served on the editorial boards for several international journals.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, the Internet of Things, 5G and beyond, and vehicular *ad hoc* and sensor networks. He is a Registered Professional Engineer of ON, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society. He served as the Technical Program Committee Chair/the Co-Chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, and IEEE Globecom'07, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the President Elect of the IEEE Communications Society. He was the Vice President for Technical & Educational Activities, the Vice President for Publications, a Member-at-Large on the Board of Governors, a member of the IEEE Fellow Selection Committee of the ComSoc, and the Chair of the Distinguished Lecturer Selection Committee. He received the R. A. Fessenden Award from IEEE, Canada, in 2019, the Award of Merit from the Federation of Chinese Canadian Professionals (ON) in 2019, the James Evans Avant Garde Award from the IEEE Vehicular Technology Society in 2018, the Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society, and the Technical Recognition Award from the Wireless Communications Technical Committee in 2019 and the AHSN Technical Committee in 2013. He has also received the Excellent Graduate Supervision Award from the University of Waterloo in 2006 and the Premier's Research Excellence Award (PREA) from ON, Canada, in 2003. He served as the Editor-in-Chief for the IEEE INTERNET OF THINGS JOURNAL, *IEEE Network*, and *IET Communications*.