Cloud-Edge-Client Collaborative Learning in Digital Twin Empowered Mobile Networks

Lindong Zhao, Shouxiang Ni, *Graduate Student Member, IEEE*, Dan Wu, and Liang Zhou, *Senior Member, IEEE*

***Abstract*— Digital twin (DT) has emerged as a key enabler for the intelligent-oriented evolution of mobile networks. With the rise of privacy concerns for enabling intelligent applications in DT-empowered mobile networks (DTMNs), federated learning has garnered wide attention due to its potential on breaking down data silos. However, the data privacy of federated learning is greatly threatened by emerging gradient leakage attacks, and the need for frequent knowledge exchange limits its training efficiency over resource-constrained DTMNs. To circumvent such dilemmas, this work first proposes a privacy-enhanced federated learning framework based on cloud-edge-client collaborations. Particularly, model splitting between clients and edge servers makes gradient leakage attacks computationally prohibitive, and cloud-side partial model aggregation provides hierarchical data utility. To improve the training efficiency of the proposed learning framework, we further establish its communication and computation cost models, and develop a DT-assisted multi-agent deep reinforcement learning-based resource scheduler for joint client association and channel assignment. Finally, as a case study of intelligent applications in DTMNs, a human-robot collaborative nursing task is designed to evaluate the practical performance of our proposed scheduler. Experimental results show its superiority in saving training costs and preserving learning accuracy.**

***Index Terms*— Digital twin, privacy-enhanced federated learning, data-driven resource allocation, human–robot collaboration.**

I. Introduction

THANKS to recent advances in edge computing and artificial intelligence (AI), mobile networks are expected to act as edge infrastructures to provide intelligent applications, instead of serving connectivity purpose alone [1]. In this context, ubiquitous sensors adaptively monitor physical environments and deliver their observations to edge servers for making decisions without human intervention [2]. In order to speed up such intelligent-oriented evolution, digital twin empowered mobile network (DTMN) emerges as a promising paradigm to connect machine intelligence and communication systems [3]. In DTMNs, digital twins (DTs) of network entities are created through software definition to represent, simulate and predict their behavior and context [4]. In essence, DT can be regarded as a set of coupled emulation models that evolve over time to persistently synchronize operational data of the asset, and utilizes such information to facilitate data-driven decision-making and asset-specific analysis [5].

Enabling intelligent applications in DTMNs require massive data collection for training deep learning models [6]. In fact, most of training data is confidential, and even distributed across different individuals or institutions. In this way, the application of centralized learning that aggregates data in a transparent manner will incur a serious risk of privacy disclosure. On the other hand, local training protects privacy by avoiding information sharing, but is in conflict with stochastic gradient descent and random batch sampling, which are the key to the success of deep learning. Accordingly, there exists a dilemma between data utilization and privacy protection.

Recently, federated learning is proposed with high hopes for resolving the above dilemma [7]. Its core idea is to run the following three steps iteratively: i) Each client trains the machine learning model with its local dataset in parallel. ii) A server fuses the locally trained models of all clients to form a global model. iii) The server distributes this global model to each client as the initialization model for the next round of local training. By transferring the computation to the curators of training data and eliminating the dependency on centrally pooled data, federated learning can improve the data confidentiality of distributed clients.

Although federating learning can provide a good level of data privacy by allowing only the distribution of model parameters, it still faces a number of critical challenges, especially in DTMNs.

Challenge 1: How to enhance privacy protection without hurting data utilization or training efficiency, especially in the face of emerging novel attacks?

The privacy protection capability of federated learning is growingly questioned by the recent emergence of *gradient leakage attack* [8], [9], [10], where the server for fusing locally trained models can extrapolate training data from the shared parameter updates. Specifically, the malicious server first randomly generates a pair of pseudo sample and pseudo label, and then performs the regular forward and back propagation to obtain pseudo gradient of model parameters. Instead of adjusting model parameters to minimize the gap between ground-truth and inference results as in regular model training, the malicious server optimizes pseudo sample-label to minimize the gap between real gradient shared from clients

Manuscript received 30 November 2022; revised 20 May 2023; accepted 4 August 2023. Date of publication 30 August 2023; date of current version 26 October 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62122094 and Grant 62231017 and in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions. *(Corresponding author: Dan Wu.)*

Lindong Zhao, Shouxiang Ni, and Liang Zhou are with the Key Laboratory of Broadband Wireless Communication and Sensor Network Technology, Nanjing University of Posts and Telecommunications, Ministry of Education, Nanjing 210003, China (e-mail: lindongzhao1996@126.com; nsx250762979@163.com; liang.zhou@njupt.edu.cn).

Dan Wu is with the Institute of Communications Engineering, Army Engineering University of PLA, Nanjing 210001, China (e-mail: wujing1958725@126.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3310060.

Digital Object Identifier 10.1109/JSAC.2023.3310060

and pseudo gradient generated by itself. In this way, pseudo samples can be made close to private training data by matching pseudo gradients with real gradients. Such a attack is even more harmful in DTMNs due to the open nature of wireless medium, *i.e.*, it can be performed via eavesdropping trusted servers in DTMNs. Given the novel attacks faced with federated learning, the existing solutions either introduce considerable computing overhead (homomorphic encryption [11]) or hurt the training performance (differential privacy [12], [13]). As such, it is necessary to enhance the data privacy of federated learning without introducing extra privacy computing burdens.

Challenge 2: How to improve training efficiency over mobile networks, especially under a desired balance between privacy protection and data utilization?

Iterative knowledge exchange (*e.g.*, gradient/parameter exchange) is inevitable for federated learning in balancing privacy protection and data utilization, but brings about a huge increase in data traffic, especially in the age of large-scale model. Poor training efficiency induced by enormous data traffic will hurt the effectiveness of time-critical intelligent applications, whose AI-driven kernels require frequent and timely updates. This issue is particularly serious in DTMNs because of the ever more severe contradiction between the scarcity of spectrum resources and the explosion of mobile traffic. In view of the stochasticity of wireless environments and the heterogeneity of mobile devices, radio resource allocation toward improving training efficiency is becoming one of the bottleneck problems for applying federated learning over DTMNs.

In response to the above challenges, this work aims to design a privacy-enhanced and spectrum-efficient federated learning scheme in DTMNs. The contributions are summarized as follows.

- To resolve Challenge 1, we refine the framework of federated learning by enabling model splitting among cloud-edge-client. The refined framework is called *cloud-edge-client collaborative learning*, where clients first execute the learnable model up to a certain layer in parallel on local datasets, and deliver the feature maps to edge servers for training the remaining model. Then, edge servers pass the gradients of those feature maps to the respective clients for completing back propagation. Finally, the cloud server aggregates client-side and edge-side models with different levels of randomness. As such, split back propagation between clients and servers makes gradient leakage attacks computationally prohibitive, and partial model aggregation on the cloud provides controllable data utilization.
- To resolve Challenge 2, we exploit the inherent emulation capability of DTMNs to generate experiences for enabling an intelligent resource scheduler. First, a joint client association and channel assignment problem is formulated to capture training costs and competitive behaviors in cloud-edge-client collaborative learning, and is transformed into a multi-agent sequential decision problem due to the exponential complexity. With the goal of jointly minimizing training time and energy cost, dueling double deep Q-network (D3QN) based agents are developed for efficient resource scheduling in the millisecond time-scale. Since real-world networks cannot afford information acquisition on global channel state and random explorations, those agents are tailored for a practically available and fingerprint-based state space, and learn from a high-fidelity network emulation constructed by DTs.
- To evaluate the practical performance of the proposed resource scheduler in cloud-edge-client collaborative learning, we design a medical human-robot collaboration task as a case study of intelligent applications in DTMNs. The experimental results demonstrate the superiority of our designed scheduler in saving training costs and preserving learning accuracy.

The remainder of this paper is organized as follows. In Section II, the related literature is reviewed. The procedure of cloud-edge-client collaborative learning is described in Section III. Next, the training efficiency of the proposed learning framework over DTMNs is analyzed in Section IV. Subsequently, a DT-assisted intelligent resource scheduler is developed in Section V. Experimental evaluation is presented in Section VI, followed by conclusions in Section VII.

II. Related Work

A. Federated Learning in Digital Twin Empowered Mobile Networks

Since the rising concerns of data privacy pose critical challenges on enabling intelligent applications in DTMNs, there has been a growing attention on exploiting federated learning over DTMNs. Specifically, the existing studies [15], [16], [17], [18], [19] mainly focus on improving the communication efficiency of federated learning over mobile networks. For example, optimizing client selection and resource allocation for balancing the tradeoff between communication overhead and learning accuracy [15], [16], [17], investigating the economic issue between DTMN operators and resource-limited clients for expanding the scale of federated learning [18], and asynchronously adjusting the frequency of federated aggregation for resolving the straggler effect [19].

Note that, the above studies ignore the risk of federated learning being subjected to novel attacks (*e.g.*, gradient leakage attack). In this context, it is natural to think about introducing homomorphic encryption and differential privacy to enhance the privacy of federated learning, such as [11], [12], and [13]. However, they either damage the training performance of federated learning or incur enormous computing costs, which serves as a motivation for our present work. Besides, the existing works of federated learning in DTMNs are typically evaluated over simple machine learning tasks such as handwriting recognition. Actually, the impact of communication efficiency on the data utilization of federated learning needs to be verified in practical application scenarios.

B. Digital Twin Assisted Intelligent Resource Allocation in Mobile Networks

There have been increasing interests in utilizing DT to facilitate AI-driven radio resource allocation, due to its ability on providing a predictive and autonomous software replica of

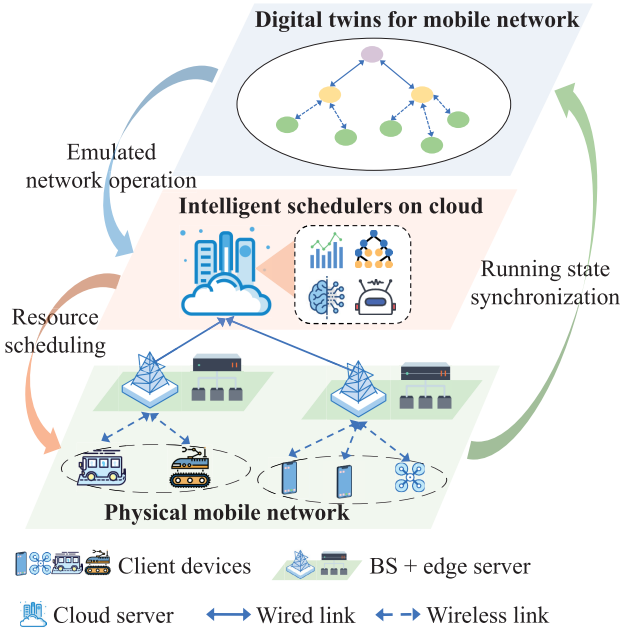


Fig. 1. Illustration of the considered digital twin empowered mobile network.

mobile networks. Particularly, the DTMNs are constructed to capture stochastic network dynamics and train deep learning based optimization solvers. Various works have discussed different application areas of the aforementioned paradigm, including ultra-reliable and low-latency communications [20], computation offloading [21], unmanned aerial vehicle networks [22], Internet of Vehicles [23], and industrial Internet of Things (IoT) [24].

Although the exciting advantages of DT-assisted intelligent resource allocation for wireless networking have been recognized, the existing designs are not suitable for our proposed cloud-edge-client collaborative learning framework. That is because, the existing DTMN modeling and scheduling schemes are not optimized for the coexistence of split back propagation, iterative feature exchange and federated aggregation. Such a coexistence is not only the key to the proposed learning framework in balancing privacy protection and data utilization, but also endogenously incurs novel resource allocation issues.

III. Privacy-Enhanced Federated Learning Under Cloud-Edge-Client Collaborations

In response to the risk of gradient leakage attacks in federated learning, this section proposes a privacy-enhanced federated learning framework based on cloud-edge-client collaborations. In what follows, the network model and learning procedure are described with formal notations.

A. Network Model

As shown in Fig. 1, we consider a DTMN consisting of three kinds of physical entities, *i.e.*, N distributed clients with sensing, communication and computation capabilities, M base stations (BSs) each of whom is equipped with an edge server, and a cloud server. Their objective is to collaboratively train an AI model for enabling intelligent applications. The clients are indexed by $\mathcal{N} = \{1, \ldots, N\}$, each of which observes

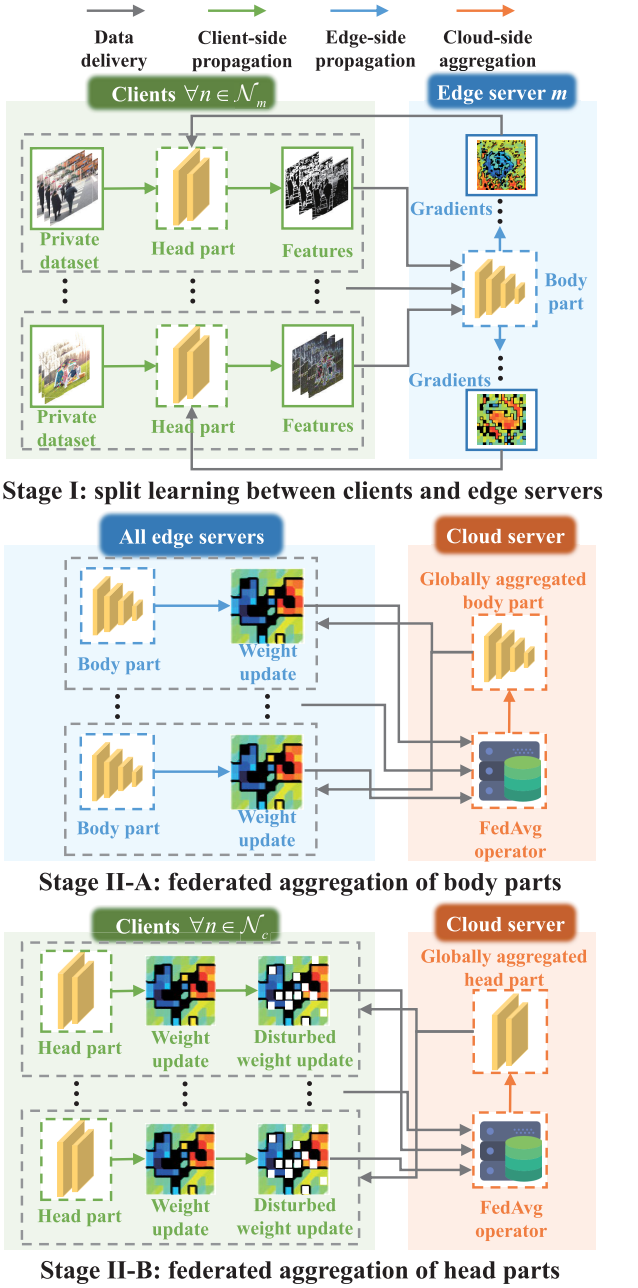


Fig. 2. Illustration of the proposed cloud-edge-client collaborative learning framework.

the physical environment from its on-device sensors and thus holds a private dataset. The edge servers $\mathcal{M} = \{1, \ldots, M\}$ are associated with the clients by sharing a set of orthogonal wireless channels $\mathcal{K} = \{1, \ldots, K\}$. The cloud server could be a centralized server connected to BSs through core networks, or one of edge servers deployed on BSs. Moreover, each client is permitted to access only one edge server through one channel.

To assist distributed learning over wireless networks, DT replicas of network entities are created to constantly synchronize their running states such as computing capability, communication capacity, and energy consumption. Those DT replicas are virtual models of network entities, and jointly establish a digital mirror of the actual network using the collected observations and estimations. Through online

monitoring or offline simulating the typologies, dynamics and properties of mobile networks, DT can provide rich information for applying intelligent resource schedulers.

B. Learning Procedure

As shown in Fig. 2, the proposed learning framework is composed of three steps: I) split learning between clients and edge servers in parallel, II-A) federated aggregation of client-side models on the cloud, and II-B) federated aggregation of edge-side models on the cloud.

In the initialization phase, the AI model $\mathbf{W}$ to be trained is divided into two portions: i) head part $\mathbf{W}^H$ (part of the AI model close to data input, *e.g.*, a shallow feature extractor) and ii) body part $\mathbf{W}^B$ (remaining layers close to inference output, *e.g.*, a deep feature extractor with a linear classifier). A complete AI model is formed by successively connecting the head part and the body part. The model details (including network structure, weights and bias) of head parts are privately determined by clients and are never shared, while those of body parts are transparently shared to the edge and cloud servers. Each client randomly initializes the parameterization of its private head part $\mathbf{W}^H_{n,0}$, and the cloud server randomly initializes the parameterization of the shared body part $\mathbf{W}^B_{m,0}$ and broadcasts it to all edge servers. After model splitting and initialization, the forward and back propagation of the AI model is done by iterations of the following two stages, where Stage II has two parallel steps.

In round r of Stage I (split learning between clients and edge servers in parallel), each client $n \in \mathcal{N}$ performs forward propagation of the head part $\mathbf{W}^H_{n,r}$ with a data batch $\mathcal{D}^r_n = \{(x^i_{n,r}, y^i_{n,r}) : i \in \{1, \ldots, D^r_n\}\}$ in its private dataset.D^r_n is the number of data samples from client n in round r, $x^i_{n,r}$ is the i-th data sample in $\mathcal{D}^r_n$, and $y^i_{n,r}$ is the ground-truth label of $x^i_{n,r}$. Then, it uploads the feature maps $\mathbf{W}^H_{n,r}(x_n)$ (also called activations or smashed data) along with the ground-truth label y_n to its associated edge server. After receiving the feature maps from the associated clients, each edge server $m \in \mathcal{M}$ completes forward propagation of its body part $\mathbf{W}^B_{m,r}$ in parallel and obtains the predicted labels $\{\mathbf{W}^B_{m,r}(\mathbf{W}^H_{n,r}(x_n))\}_{n\in\mathcal{N}_m}$, where $\mathcal{N}_m$ denotes the set of clients connected with edge server m. At this point, a single forward path is completed.

Next, the back propagation for the body part deployed on edge server $m \in \mathcal{M}$ can be performed as:

$$\min_{\mathbf{W}^B_m} \sum_{n\in\mathcal{N}_m} \sum_{i=1}^{D^r_n} \mathcal{L}(y^i_{n,r}, \mathbf{W}^B_{m,r}(\mathbf{W}^H_{n,r}(x^i_{n,r}))), \tag{1}$$

where $\mathcal{L}(y, \hat{y})$ denotes the task-specific loss function between the ground-truth y and predicted label $\hat{y}$. After carrying out parameter update of the body part, each edge server delivers the gradients of the feature maps to respective clients. With the gradients of those feature maps, client $n \in \mathcal{N}$ performs back propagation for its private head part by solving the following optimization problem:

$$\min_{\mathbf{W}^H_n} \sum_{i=1}^{D^r_n} \mathcal{L}(y^i_{n,r}, \mathbf{W}^B_{m,r}(\mathbf{W}^H_{n,r}(x^i_{n,r}))). \tag{2}$$

That completes a single backward path.

In round r of Stage II-A (federated aggregation of body parts of all clients on the cloud), each edge server sends its current body parameters $\mathbf{W}^B_{m,r}$ obtained in Stage I to the cloud server. Subsequently, the cloud aggregates the edge-side parameter updates and feeds the resulting body part back to each edge server as an initialization model for the next round:

$$\mathbf{W}^B_{m,r+1} = \frac{\sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}_m} D^r_n \mathbf{W}^B_m}{\sum_{n\in\mathcal{N}} D^r_n}, \forall m \in \mathcal{M}. \tag{3}$$

Since the computations for federated aggregation are not costly, the role of the cloud server can also be acted by any edge server. Note that the model aggregation process is algorithm-dependent. Except for federated averaging (FedAvg) shown in eq. (3), there also exist other parameter aggregation algorithms [14].

In round r of Stage II-B (federated aggregation of head parts of certain clients on the cloud), if a client aims to maximize model training performance, it could deliver the current parameterization of its head part to the cloud server for further aggregation. Since the cloud server could be a malicious attacker, differential privacy technologies need to be incorporated to weaken its ability to extrapolate training data from weight updates. The above process can be represented as:

$$\mathbf{W}^H_{n,r+1} = \begin{cases} \frac{\sum_{n\in\mathcal{N}_C} D^r_n \hat{\mathbf{W}}^H_n}{\sum_{n\in\mathcal{N}_C} D^r_n}, \forall n \in \mathcal{N}_C, \\ \mathbf{W}^H_n, \forall n \in \mathcal{N}/\mathcal{N}_C, \end{cases} \tag{4}$$

where $\hat{\mathbf{W}}^H_{n,r}$ is the head part parameters after disturbance, and $\mathcal{N}_C$ is the set of clients participating in head part aggregation. In addition, Stage II-B is unnecessary for clients who prefer privacy protection than data utilization.

The details of cloud-edge-client collaborative learning are also summarized in Algorithm 1. Overall, it owns the following merits. First, it offers better data privacy than traditional federated learning due to model splitting. That is, the curious servers or eavesdroppers have access only to the parameterization of the body part and feature maps, while the parameter setting of the head part privately determined by a client is never shared and may be different from the counterparts of other clients. For performing gradient leakage attacks, attackers have to accurately infer all client-dependent parameters (*e.g.*, network structure, layer type and even the weights and bias) of the head part or generate pseudo gradient with every possible client-side parameter setting. The possibility to infer the client-side model parameters is highly unlikely if we allow the fully connected layers with sufficiently large nodes at the client-side model. On the other hand, the computational complexity of performing gradient leakage attacks via enumerating all client-side parameter settings increases exponentially with the parameter size. Therefore, model splitting makes gradient leakage attacks mathematically challenging and computationally prohibitive. Second, the proposed framework is more friendly to budget-strapped IoT devices, since it shifts part of client-side computing loads to edge servers. In other words, it assigns only the first few layers of the AI model to clients for training (compared to assigning the complete model as in

Algorithm 1 Cloud-Edge-Client Collaborative Learning

Split the AI model $\mathbf{W}$ to be trained into head part $\mathbf{W}^H$ and body part $\mathbf{W}^B$
Initialize body weights $\mathbf{W}_0^B$ for edge servers and head weights $\mathbf{W}_0^H$ for clients
for each round $r = 1, \ldots, R$ **do**
  if $r = 1$ **then**
    $\mathbf{W}_{m,r}^B \leftarrow \mathbf{W}_0^B, \forall m \in \mathcal{M}$
    $\mathbf{W}_{n,r}^H \leftarrow \mathbf{W}_0^H, \forall n \in \mathcal{N}$
  end if
  // *Stage I*:
  for each edge server $m \in \mathcal{M}$ **in parallel do**
    for each client $n \in \mathcal{N}_m$ **in parallel do**
      $(x_n, y_n) \leftarrow$ current data batch from n
      Client n calculates $\mathbf{W}_{n,r}^H(x_n)$ and sends it with y_n to edge server m
      Edge server m calculates $\mathbf{W}_{m,r}^B(\mathbf{W}_{n,r}^H(x_n))$
      Edge server m calculates $\frac{\partial \mathcal{L}(y_n, \mathbf{W}_{m,r}^B(\mathbf{W}_{n,r}^H(x_n)))}{\partial \mathbf{W}_{n,r}^H(x_n)}$ and sends it to client n
      Client n updates its head part by solving (2)
    end for
    Edge server m updates its body part by solving (1)
  end for
  // *Stage II*:
  for each $j \in \mathcal{M} \cup \mathcal{N}_C$ **in parallel do**
    if $j \in \mathcal{M}$ **then**
      Edge server j sends $\mathbf{W}_{j,r}^B$ to the cloud //*Stage II-A*
    else
      Client j sends $\mathbf{W}_{j,r}^H$ to the cloud //*Stage II-B*
    end if
    Federated aggregation of body parts as in (3)
    Federated aggregation of head parts as in (4)
  end for
end for

traditional federated learning). Third, it achieves hierarchical and controllable data utilization by allowing partial model aggregation under differential privacy. Clients who pursuit data utility maximization could transparently share the weights of its private model, while others with varying degrees of privacy requirements could resort to imposing random noises.

IV. Training Efficiency of Cloud-Edge-Client Collaborative Learning

Given the training efficiency issue induced by frequent knowledge exchange in federated learning, this section firstly analyzes the computation and communication costs of the proposed learning framework in DTMNs. Guided by the cost models, a joint client association and channel assignment problem is then formulated.

A. Analysis of Computation Costs

The computation costs of the proposed learning framework mainly derive from client-side forward and back propagation of the head part, and edge-side counterparts of the body part. Since the computation for federated aggregation is not costly, the corresponding costs are ignored.

Let F_n^H denote the computing capability (floating point operations per second, FLOPS) per sample of client n for performing forward and back propagation of the head part, and ω_n^H represent the energy consumption per sample of client n, whose values depend on client-side computation hardware. ψ^H is the computing load (floating point operations, FLOPs) per sample of client n, which is client-independent. Accordingly, the local computing delay of client n in round r can be calculated as

$$T_{n,r}^{H,cp} = \frac{D_n^r \psi^H}{F_n^H}. \tag{5}$$

And the energy consumption of client n in round r for local computing can be expressed as

$$E_{n,r}^{H,cp} = D_n^r \omega_n^H. \tag{6}$$

Let F_m^B and ψ^B be the computing capability of edge server m and the computing load per sample for completing forward and back propagation of the body part. The edge computing delay of client n in round r can be expressed as

$$T_{n,r}^{B,cp}(\boldsymbol{\alpha}) = \sum_{m \in \mathcal{M}} \frac{\alpha_{n,m} D_n^r \psi^B}{F_m^B (D_n^r / \sum_{n' \in \mathcal{N}} \alpha_{n',m} D_{n'}^r)}, \tag{7}$$

where $\boldsymbol{\alpha} = [\alpha_{n,m}]_{N \times M}$ is a binary client association matrix and $\alpha_{n,m}$ is denoted as

$$\alpha_{n,m} = \begin{cases} 1, \text{client } n \text{ accesses to server } m, \\ 0, \text{otherwise.} \end{cases} \tag{8}$$

Eq. (7) models the competition toward finite edge computing resources by incorporating multi-threading and multi-processing instead of fixing quotas, *i.e.*, edge-side computing power are equally allocated to the received samples. Considering that edge servers are typically connected to fixed power supplies, the energy consumption for edge computing is not considered in this work.

B. Analysis of Communication Costs

The communication overhead of the proposed learning framework is dominated by iterative feature/gradient exchange. As federated aggregation of head parts is not necessary for all clients, the communication overhead of Stage II-B is not incorporated. Moreover, the costs of wired transmission for federated aggregation of body parts are not considered.

The uplink communication capacity achieved by client n in round r for feature delivery can be written as

$$R_{n,r}^H(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{m \in \mathcal{M}} \sum_{k \in \mathcal{K}} G \log_2(1 + \Gamma_{n,m}^{H,k}), \tag{9}$$

where G is the channel bandwidth. $\Gamma_{n,m}^{H,k}$ denotes the uplink signal-to-interference-and-noise ratio (SINR) of client n associated with edge server m on channel k, and is written as

$$\Gamma_{n,m}^{H,k} = \frac{\alpha_{n,m} \beta_{n,k} P_n h_{n,m}^k}{\sigma^2 + \sum_{n' \in \mathcal{N} \setminus \{n\}} \beta_{n',k} h_{n',m}^k P_{n'}}, \tag{10}$$

where $h_{n,m}^{k}$ is the channel gain on channel k from client n to edge server m, σ^2 is the variance of Gaussian noise, P_n is the transmission power of client n, and $\boldsymbol{\beta} = [\beta_{n,k}]_{N\times K}$ is a binary channel allocation matrix where $\beta_{n,k}$ is written as

$$\beta_{n,k} = \begin{cases} 1, \text{client } n \text{ accesses to channel } k, \\ 0, \text{otherwise.} \end{cases} \tag{11}$$

Accordingly, the uplink transmission delay of client n in round r, is expressed as

$$T_{n,r}^{H,cm}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \frac{D_n^r \xi^H}{R_{n,r}^H}, \tag{12}$$

where ξ^H is the size of the feature maps per sample. The energy consumption of client n in round r for uplink transmission can be expressed as

$$E_{n,r}^{H,cm}(\boldsymbol{\alpha},\boldsymbol{\beta}) = P_n \frac{D_n^r \xi^H}{R_{n,r}^H}. \tag{13}$$

The downlink transmission rate achieved by client n in round r can be expressed as

$$R_{n,r}^{B}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \sum_{m\in\mathcal{M}}\sum_{k\in\mathcal{K}} G\log_2(1+\Gamma_{m,n}^{B,k}), \tag{14}$$

where $\Gamma_{m,n}^{B,k}$ denotes the downlink SINR of client n associated with edge server m on channel k, and is written as

$$\Gamma_{m,n}^{B,k} = \frac{\alpha_{n,m}\beta_{n,k}P_m h_{m,n}^k}{\sigma^2 + \sum\limits_{m'\in\mathcal{M},n'\in\mathcal{N}\setminus\{n\}} \alpha_{n',m'}\beta_{n',k}h_{m',n}^k P_{m'}}. \tag{15}$$

The time client n takes to receive the gradients of feature maps, is denoted as

$$T_{n,r}^{B,cm}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \frac{D_n^r \xi^B}{R_{n,r}^B}, \tag{16}$$

where ξ^B is the size of the gradients of feature maps per sample. Note that the energy consumption of wireless infrastructures for downlink transmission is also ignored.

According to the modeling of communication and computation costs, if too many clients associate with the same edge server for split learning or access the same channel for knowledge exchange, strong congestion effects (co-channel interference or server overload) will incur unacceptable energy and time costs. In this regard, client associations $\boldsymbol{\alpha}$ and channel assignments $\boldsymbol{\beta}$ directly control the DTMN's congestion level, which dominates the training efficiency of distributed learning over DTMNs. Hence, the joint client association and channel assignment problem should be investigated to improve the training efficiency of the proposed learning framework in DTMNs.

C. Problem Formulation

Due to the necessary for federated aggregation of body parts, the total training time of the proposed learning framework in round r can be denoted as

$$T_r(\boldsymbol{\alpha},\boldsymbol{\beta}) = \max_{n\in\mathcal{N}}(T_{n,r}^{H,cp} + T_{n,r}^{B,cp} + T_{n,r}^{H,cm} + T_{n,r}^{B,cm}). \tag{17}$$

The total energy consumption of client n for participating in round r of the proposed learning framework is written as

$$E_{n,r}(\boldsymbol{\alpha},\boldsymbol{\beta}) = E_{n,r}^{H,cp} + E_{n,r}^{H,cm}. \tag{18}$$

With the goal of minimizing the system cost (a weighted sum of training time and energy consumption), the optimization problem is formulated as follows.

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \lambda\sum_{r=1}^{R} T_r(\boldsymbol{\alpha},\boldsymbol{\beta}) + \mu\sum_{r=1}^{R}\sum_{n=1}^{N} E_{n,r}(\boldsymbol{\alpha},\boldsymbol{\beta}) \tag{19a}$$

$$\text{s.t.} \quad C1: \sum_{r=1}^{R} T_r \le T^{\max}, \tag{19b}$$

$$C2: \sum_{r=1}^{R} E_{n,r} \le E_n^{\max}, \forall n\in\mathcal{N}, \tag{19c}$$

$$C3: \sum_{m\in\mathcal{M}} \alpha_{n,m} \le 1, \forall n\in\mathcal{N}, \tag{19d}$$

$$C4: \sum_{k\in\mathcal{K}} \beta_{n,k} \le 1, \forall n\in\mathcal{N}, \tag{19e}$$

$$C5: \sum_{n\in\mathcal{N}} \alpha_{n,m}\beta_{n,k} \le 1, \forall m\in\mathcal{M}, \forall k\in\mathcal{K}, \tag{19f}$$

$$C6: \alpha_{n,m}, \beta_{n,k} \in \{0,1\}, \forall n,m,k. \tag{19g}$$

where $T^{\max}$ in (19b) is the latency budget for model training, $E_n^{\max}$ in (19c) is the energy budget of client n, and the constrains (19d)-g) ensure the one-to-one matching between clients and server-channel pairs. $\lambda, \mu \in [0,1]$ are weighting coefficients of latency and energy consumption for providing rich modeling flexibility toward the considered DTMN. In practice, the above elasticities could be determined via multi-criteria utility theory [25].

Referring to previous works on radio resource allocation in DTMNs [20], [21], [22], [23], [24], a quasi-static network/channel assumption is adopted, *i.e.*, the client locations and radio environments remain static during a round, but independently change across rounds. Therefore, resource allocation could be made at the beginning of a round and remain effective within this round. Despite the above simplified assumption, solving (19) with traditional centralized optimizations are still mathematically challenging and computationally prohibitive. For instance, the computational overhead of searching (19) within a round via a brute force method is proportional to $(M\times K)^N$. Since this kind of integer programming could degenerate into a NP-hard multiple-knapsack problem [26], it cannot be optimally searched via any known polynomial method.

V. DT-Assisted Intelligent Resource Allocation for Cloud-Edge-Client Collaborative Learning

In view of the exponential complexity of the joint client association and channel assignment for the proposed learning framework, this section resorts to DT-assisted deep reinforcement learning (DRL) to enable data-driven resource allocation. We first transform the optimization problem in (19) into a Markov decision process (MDP), and then propose a multi-agent DRL-based resource scheduler.

A. Markov Decision Process

In this section, we regard each client as an agent.[1] All agents continuously interact with the DT environment (virtual networks emulated by DT) and refine resource requests according to their own experiences. The multi-agent setting is adopted due to the curse of dimensionality. Specifically, the input and output cardinality of a learnable kernel for solving problem (19) in a centralized DRL manner is proportional to $(M \times K)^N$. Training such a kernel is extremely hard, since performing random exploration necessary for DRL in a exponentially growing state-action space is highly inefficient.

Mathematically, an MDP describing trail-and-error interactions between agents and the DT environment can be modeled as follows. In round r, each agent $n \in \mathcal{N}$ performs action $a_n^r \in \mathcal{A}$ with local state $s_n^r \in \mathcal{S}$ observed from the DT environment, according to its own resource request policy. The actions of all agents collectively form an action vector, which affects the DT environment and evolves it to a new status. Then, each agent receives an immediate reward $U_n^r(\{a_n^r\}_{n\in\mathcal{N}}, \{s_n^r\}_{n\in\mathcal{N}})$ capturing the impact of the joint action, and observes new local state $s_n^{r+1} \in \mathcal{S}$ reflecting the evolved DT environment. Performing the above process in an iterative manner forms a set of experiences $\{s_n^r, a_n^r, U_n^r, s_n^{r+1}\}_{r=1,2,\ldots}$, which will be used by agent n to update its individual resource request policy for maximizing the cumulative reward.

In the following, the instantiations of key elements of the considered MDP are provided.

1) State: The dimension of a complete DT environment is more than $(M \times K)^N$ due to the involvement of global channel state information, device location, and so on. As an agent with fast convergence and efficient exploration demands cannot afford a high-dimensional state space, it is important to select essential environment representations. Moreover, each agent faces a non-stationary cumulative reward function in our considered MDP, since the policy updates of other agents also influence the environment evolution. Such non-stationarity greatly hurts the performance of experience replay and thus destabilizes the training convergence, where randomly selected experiences cannot capture current environment changes. In order to solve the above issue, a fingerprint-based paradigm [27] is introduced into our state space design. The basic idea is, incorporating an estimation toward the policies of other agents into local state observation can approximately make the cumulative reward function stationary. Considering that the policy of a DRL agent is typically composed of a deep neural network (DNN) with numerous parameters, a lightweight fingerprint reflecting joint policies is needed.

As a result, the local state observed by agent n is defined to contain

- $\{|\mathcal{N}_m^{r-1}|\}_{m\in\mathcal{M}} \cup \{|\mathcal{N}_k^{r-1}|\}_{k\in\mathcal{K}}$, the number of clients associating with each server and channel, which reflects the network congestion levels (also negative network externality) controlled by joint policies;
- $\{\sum_{r'=1}^{r-1} E_{n,r'}/E_n^{\max}, \sum_{r'=1}^{r-1} T_{r'}/T^{\max}\}$, the consumption ratio of energy and latency budget.

[1]In Section V, we use the term "client" as being interchangeable with "agent".

2) Action: As each client needs to associate with an edge server and reuse a channel per round, the action space of agent n can be denoted as $\mathcal{A} = \{(\alpha_{n,m}, \beta_{n,k})\}_{m\in\mathcal{M},k\in\mathcal{K}}$. The dimension of this action space is $M \times K$, where each action is equivalent to a joint client association and channel assignment.

3) Reward: In terms of the joint client association and channel assignment problem (19) for the proposed learning framework, the objective has twofold: maximizing the number of training rounds for better data utilization, while reducing the time and energy consumption during model training. Consequently, we incorporate the reciprocal of network costs per round into the immediate reward, and set a constant reward per round until the energy and latency budget is exhausted. Moreover, the same reward function is used for all agents to avoid the resource allocation issue transforming into a competitive game. Formally, the immediate reward in round r for agent n is represented as

$$U_n^r = \begin{cases} 0, \sum_{r'=1}^{r} T_{r'} > T^{\max} \text{ or } \sum_{r'=1}^{r} E_{n,r'} > E_n^{\max}, \\ \rho + 1/(\lambda T_r + \mu \sum_{n=1}^{N} E_{n,r}), \text{ otherwise.} \end{cases} \tag{20}$$

Practically, the constant reward ρ is a hyperparameter whose fine-tuning will be further discussed in the experimental section.

B. Multi-Agent DRL-Based Resource Scheduler

In the above MDP problem, each agent owns an action-value function as follows,

$$Q_n(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t U_{r+t} | s_n^r = s, a_n^r = a\right], \tag{21}$$

where $\mathbb{E}[\cdot]$ is the expectation function and $\gamma \in [0,1]$ is a discount factor indicating the impact of future rewards. When the scale of the state-action space is small, independent Q-learning can approximate optimal policies for each agent by maximizing the above function [28]. Nevertheless, due to its expensive computing and caching overhead in the high-dimensional state-action space, traditional Q-learning is not applicable to our considered MDP. Consequently, this subsection develops a DRL-based resource scheduler implemented by the dueling double deep Q-network (D3QN) architecture [29], [30], [31].

As illustrated in Fig. 3, each agent n uses a dedicated DNN approximator $Q_n(s,a;\mathbf{W}_n^{RL})$ with weights $\mathbf{W}_n^{RL}$ (called online network) to estimate its optimal action-value function. As an extension of the traditional deep Q-network (DQN) method, D3QN incorporates a dueling architecture into the DNN approximator, whose output layer is split into two subnetworks with different functionalities [31]. In our design, the subnetwork outputting a scalar evaluates environment state, while another subnetwork outputting a vector with dimension $M \times K$ describes the advantage of an action over the other choices. Through combining the above subnetworks, the gain of any action taken at the current state can be estimated.

The training procedure of the proposed multi-agent DRL-based resource scheduler is specified in Algorithm 2. Each

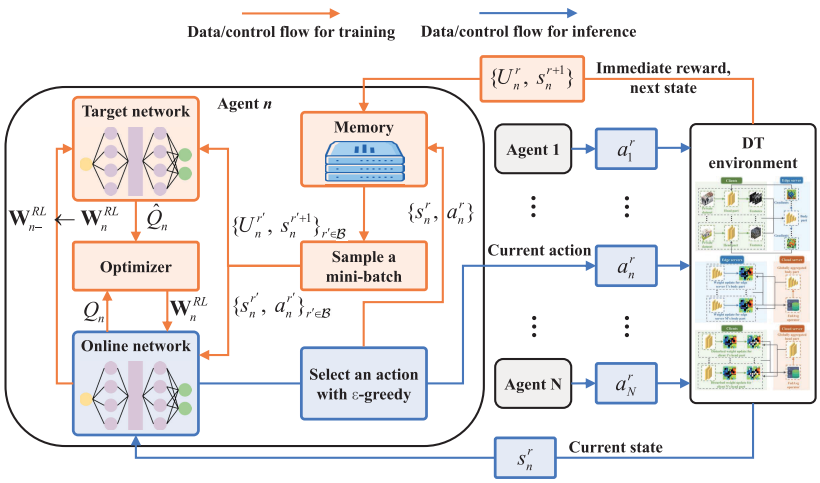


Fig. 3. Illustration of the proposed multi-agent DRL-based resource scheduler for joint client association and channel assignment in cloud-edge-client collaborative learning (take agent n as an example).

Algorithm 2 DT-Assisted Training Procedure of Multi-Agent DRL-Based Resource Scheduler for Joint Client Association and Channel Assignment in Cloud-Edge-Client Collaborative Learning

Initialize the replay memory, online network $\mathbf{W}_n^{RL}$, and target network $\mathbf{W}_{n-}^{RL}$ for each agent $n \in \mathcal{N}$
for each episode $1, 2, \ldots,$ **do**
  Randomly instantiate a DT of the considered network
  for each round $r = 1, \ldots, R$ **do**
    for each agent $n \in \mathcal{N}$ **in parallel do**
      n chooses action a_n^r based on $Q_n(s^r, a; \mathbf{W}_n^{RL})$ and ε-greedy
    end for
    Each agent $n \in \mathcal{N}$ receives an immediate reward U_n^r, and the DT environment is updated
    for each agent $n \in \mathcal{N}$ **in parallel do**
      n observes next local DT environment state s_n^{r+1}
      n stores the experience $\{s_n^r, a_n^r, U_n^r, s_n^{r+1}\}$
    end for
  end for
  if $r \bmod r^o = 0$ **then**
    for each agent $n \in \mathcal{N}$ **in parallel do**
      n samples a mini-batch from the stored experiences
      n updates the online network $\mathbf{W}_n^{RL}$ according to (22)-(23)
    end for
  end if
  if $r \bmod r^t = 0$ **then**
    for each agent $n \in \mathcal{N}$ **in parallel do**
      n updates the target network by $\mathbf{W}_{n-}^{RL} \leftarrow \mathbf{W}_n^{RL}$
    end for
  end if
end for

agent $n \in \mathcal{N}$ continuously generates and stores experiences $\{s_n^r, a_n^r, U_n^r, s_n^{r+1}\}_{r=1,2,\ldots}$ through its online network and the ε-greedy method. Namely, it selects the action $\arg\max Q_n(s, a; \mathbf{W}_n^{RL})$ with probability $1 - \varepsilon$ or a random action with probability ε. Every few steps, the online network samples a mini-batch $\mathcal{B}$ of the stored experiences, which is the so-called experience replay mechanism. Experience replay aims to reduce the correlations between training samples and avoid the DNN approximator falling into a local optimum [29]. Next, the online network updates its weights as follows,

$$\min_{\mathbf{W}_n^{RL}} \sum_{r \in \mathcal{B}} \left(\hat{Q}_n - Q_n(s_n^r, a_n^r; \mathbf{W}_n^{RL}) \right)^2. \tag{22}$$

In (22), $\hat{Q}_n$ is a target value function refined by [30] to resolve the over-optimistic estimation issue of the traditional DQN method, and is formally expressed as

$$\hat{Q}_n = U_n^r + \gamma Q_n(s_n^{r+1}, a^*; \mathbf{W}_{n-}^{RL}), \tag{23}$$

where $a^* = \arg\max_{a \in \mathcal{A}} Q_n(s_n^{r+1}, a; \mathbf{W}_n^{RL})$. Particularly, (23) is predicted by a duplicate of the online network (referred as target network $\mathbf{W}_{n-}^{RL}$) for the sake of learning stability. The weights of the target network are periodically frozen, *i.e.*, they are only updated after the back propagation of the online network has been performed several times.

Before leaving this section, it is worth mentioning the benefits of exploiting DT in DRL-based radio resource allocation. DT enables a centralized training and distributed implementation paradigm for data-driven schedulers by providing an emulated network replica, when real networks cannot provide enough training data or cannot afford to online exploration. Through exploiting the ability of DT on online monitoring or offline emulating the operational phases of mobile networks, the learnable kernel of a DRL agent can be trained offline over various radio environments and network typologies. In terms of implementation, clients could perform sequential resource requests using pre-trained kernels in the millisecond timescale. The pre-trained kernel needs to be adjusted only if there is a significant deviation between the real network and its digital replica. Since DT can estimate the performance of mobile networks without acquiring the deployment details of

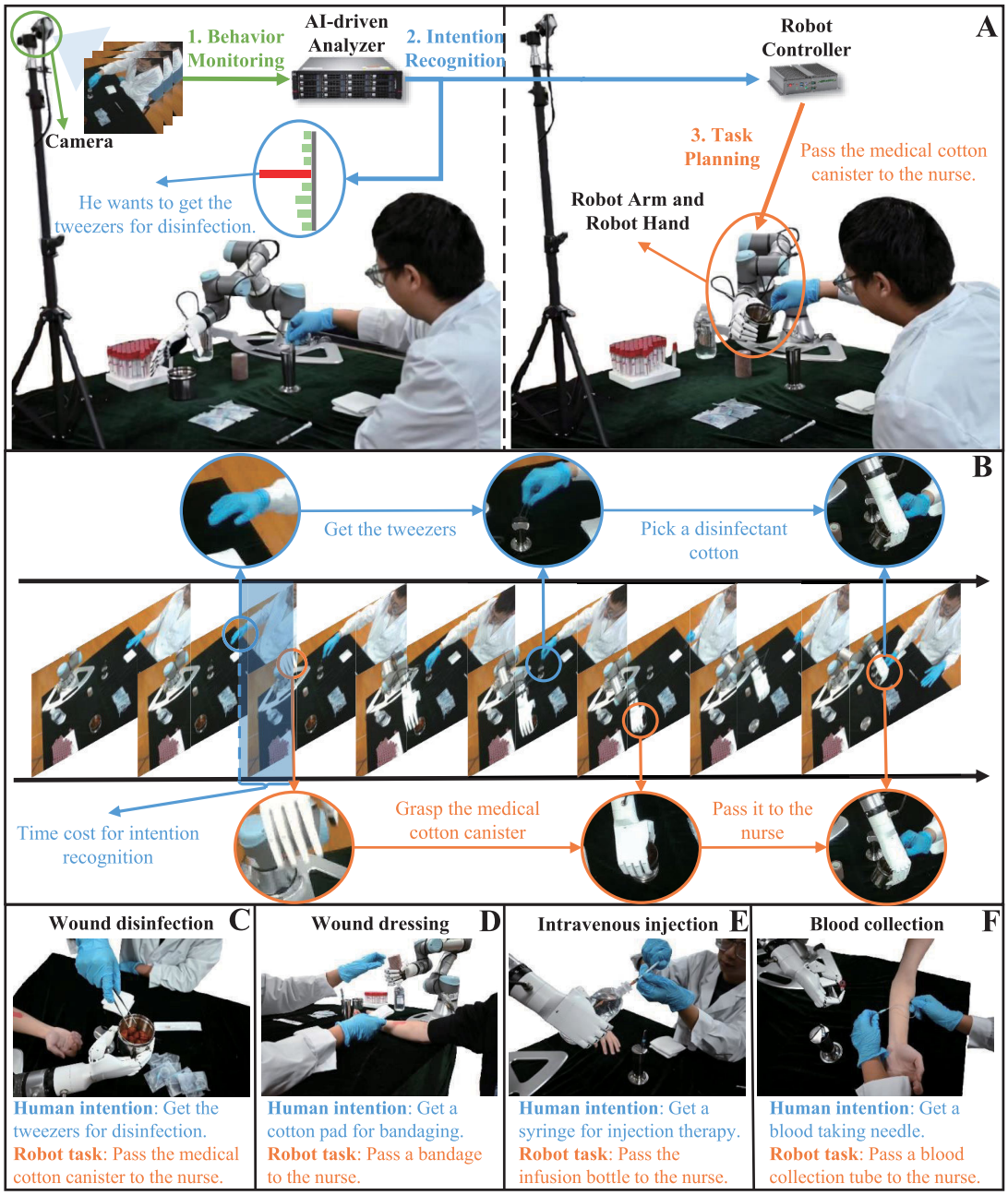


Fig. 4. Illustration of the designed human-robot collaborative nursing task: (A) core steps, (B) working example, (C-F) possible cases.

physical entities [5], it facilitates the continuous evolution of DRL agents at low signaling overhead.

VI. Performance Evaluation

To investigate the real-world impact of resource allocation on cloud-edge-client collaborative learning, this section conducts a real-world case study of intelligent applications in DTMNs. The experimental setup is first introduced, and the numerical results are subsequently analyzed.

A. Experimental Setup

The healthcare-oriented human-robot collaboration is one of the representatives of intelligent applications in DTMNs [4], [6]. In this context, we design a human-robot collaborative nursing task as the case study, whose implementation details include the following three steps as shown in Fig. 4(A-B): i) *Visual behavior monitoring*: The early action of the nurse (*e.g.*, approaching a pair of tweezers), is captured by a camera in real time. ii) *Human intention recognition*: The video streaming from the camera is synchronously analyzed by a deep learning model to recognize the nurse's intention (*e.g.*, using the tweezers for wound disinfection). iii) *Robot task planning*: Based on the intention recognition result, the robot completes the corresponding task to assist the medical treatment (*e.g.*, passing the medical cotton canister to the nurse). In a word, the AI-driven robot needs to recognize which cases shown in Fig. 4(C-F) meet the nurse's current intention and act accordingly. Given fixed task planning configuration, the robotic capability is dominated by the training performance

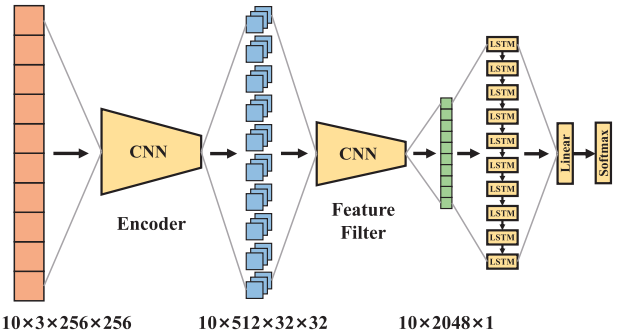


Fig. 5. Illustration of the intention recognition model to be trained.

TABLE I
RADIO ENVIRONMENT PARAMETERS AND DRL HYPERPARAMETERS

Parameter	Value
K	10
G	4MHz
$P_m,\ \forall m \in \mathcal{M}$	33dBm
$P_n,\ \forall n \in \mathcal{N}$	23dBm
σ^2	-114dBm
Path loss model	$35.3 + 37.6\log_{10}(\Delta_{n,m})$
Fast fading model	Rayleigh fading
Replay memory size	2^{17}
Exploration rate	$1 \rightarrow 0.1$
Discount factor	0.99

of the intention recognition model in the proposed learning framework.

The deep learning model involved for intention recognition is illustrated in Fig. 5. Specifically, the first three Convolutional Neural Network (CNN) based blocks of ResNet50 [32] are pre-trained with the ImageNet dataset, and jointly act as an encoder to extract the spatial features. Then, the last CNN-based block of ResNet50 is trained with the local datasets for filtering out the most critical features. Subsequently, the filtered features belonging to 10 consecutive frames are processed by LSTM [33] for temporal analysis. Finally, the spatio-temporal features are fused via a linear layer, and the fusion is processed by a softmax classifier to output recognition results. Note that the above intention recognition model is a single-view version of our another work [35] about cross-view intention recognition, and the experimental setup of human-robot collaboration also refers to it. For evaluating the training performance of this AI model, the classification accuracy (the statistical proportion of correct robotic grasping) is used as the major metric. Moreover, we collect 51200 video segments as the local dataset, which are uniformly distributed and horizontally partitioned among all clients. In the following experiments, the considered intention recognition model is trained in the proposed learning framework with different resource schedulers. Particularly, the encoder is regarded as the head part, while the rest layers jointly serve as the body part.

In terms of the mobile network configuration, we assume that $N = 16$ clients are randomly located in a 600m $\times$ 600m square, and edge servers are located at its corners (*i.e.*, $M = 4$). For the computation settings, each client and each edge server is considered to be equipped with a single NVIDA GeForce RTX 3090 GPU. Since client-side parts of the intention recognition model are set to be frozen and client-side devices are considered to be homogeneous, local computing delay and energy consumption are not incorporated in the experiments. The other network configurations are listed in Table I. As for the hyperparameters of the proposed DRL-based resource scheduler, the learnable kernel of each agent consists of three hidden layers with 512, 256, 128 neurons respectively, where the ReLU function is selected as the activation function. Moreover, the value of the constant reward ρ is fine-tuned by executing a small batch of random explorations, and is slightly greater than the reciprocal of minimum system cost per round in this batch. After that, each kernel is trained by the RMSProp optimizer [34] with a learning rate of 0.001. The other hyperparameters are also shown in Table I.

To demonstrate the effectiveness of our proposed resource scheduler, three state-of-the-art distributed resource allocation schemes in wireless networking are simulated as baselines. 1) A swap-matching based iterative algorithm, which can achieve pairwise stability for joint device association and channel assignment under the existence of congestion effects [36]; 2) a greedy search based method, which attempts to maximize the additional utility in an iterative manner [37]; 3) a sub-problem decomposition based scheme, which heuristically provides user association before channel assignment [38].

B. Numerical Results

Fig. 6(a-c) illustrate the single-round system cost, training time, and energy consumption of the proposed learning framework under different resource schedulers, along with their computational execution time for resource allocation (shown in logarithmic scale). Each colored dot indicates the training cost in a round with a specific scheduler, where the DTMN is randomly initialized with a quasi-static assumption. Specifically, all clients jointly train the intention recognition model for 2500 rounds (also 100 epochs) with a specific resource scheduler in the proposed learning framework. In the above experiments, the network operator is assumed to attach equal importance to training time and energy costs, while the latency and energy budgets for all clients are considered to be sufficient. Following the same experimental configuration as Fig. 6, Fig. 7(a-c) compare the sequential performance of different resource schedulers in terms of the system cost, training time, and energy consumption, respectively.

From the results shown in Fig. 6 and Fig. 7, we can see that the proposed scheduler achieves almost the same scheduling performance as the swap-matching based method, but reaps an overwhelming gain on reducing computational execution time. Such a fast scheduling speed guarantees the timeliness of radio resource allocation, since the time-consuming scheduler typically fails to work under the practically time-varying channel conditions. The advantage of the proposed scheduler in terms of execution time derives from the fast inference speed of the trained DNN-based DRL kernel with an aggressively cropped state space. Moreover, the proposed scheduler is obviously superior to the greedy search and subproblem

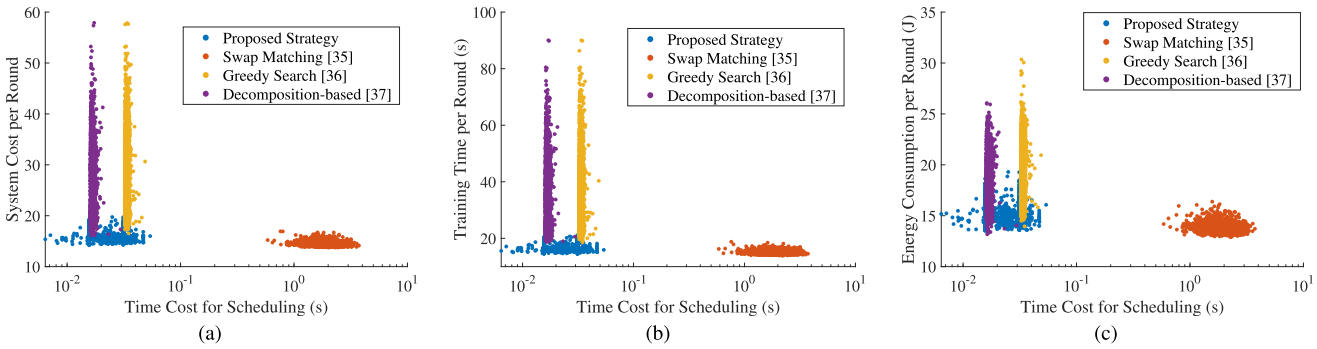


Fig. 6. Single-round performance comparison of different resource schedulers along with their computational execution time: a) weighted system cost, b) training time, c) energy consumption.

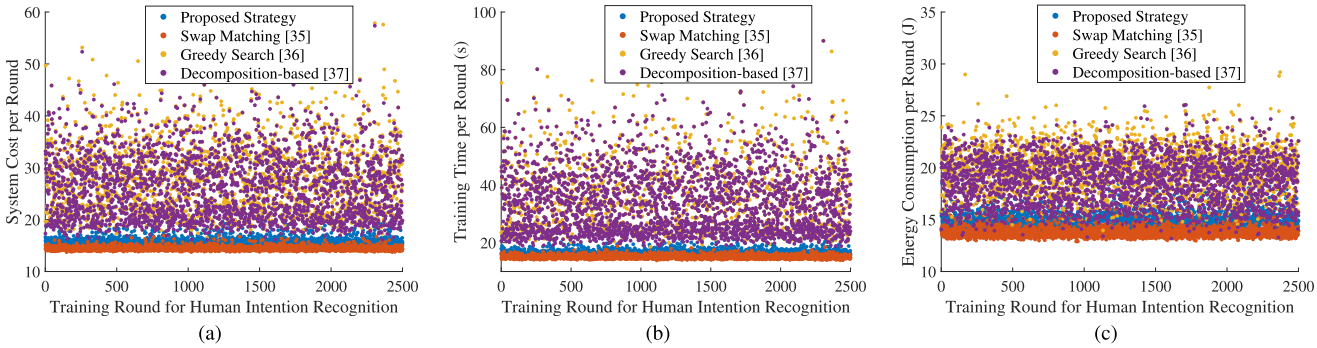


Fig. 7. Sequential performance comparison of different resource schedulers in terms of a) weighted system cost, b) training time, c) energy consumption.

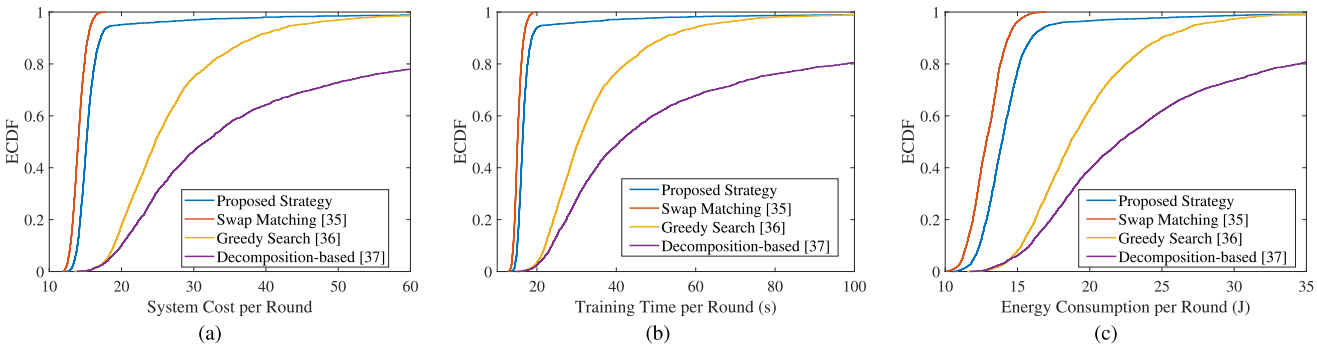


Fig. 8. ECDFs of a) weighted system cost, b) training time, c) energy consumption achieved by different resource schedulers per round.

decomposition based baselines, where the cost-saving gains come from the continuous learning of DRL agents in heterogeneous DT emulations. On the other hand, the slight inferiority of the proposed scheduler in cost-saving compared to state-of-the-art schemes such as the swap-matching based one can be briefly analyzed as follows. Although the DRL agents can learn to perceive inherent congestion effects of DTMNs and conduct effective load balancing, the aggressively cropped state space cannot capture full knowledge about dynamic radio environments such as fast channel fading.

Next, we evaluate the training cost of the proposed learning framework with different schedulers from a perspective of statistics, where Fig. 8(a-c) depict the empirical cumulative distribution function (ECDF) of the system cost, training time and energy consumption per round among different methods. The figures show that the proposed scheduler steadily outperforms the greedy search and subproblem decomposition based baselines. However, the proposed scheduler is slightly less stable than the swap-matching based method. That is because, the robustness of DRL-based methods is typically challenged by the over-fitting issue, and the exact parameterization with good generalization is hard to be fine-tuned in practice. Moreover, the observations in this section are also highly related to the particular hyperparameter setting, and extra caution is necessary for generalization.

As the training convergence of data-driven schedulers is highly dominated by the hyperparameter configuration, it is hard to investigate the convergence behavior of our proposed scheduler with an analytical method. Consequently, the convergence analysis is replaced by illustrating the loss curves of the learnable kernels of DRL agents, which is also adopted in the similar literature [19], [20]. Fig. 9 depicts the training process of four DRL agents, where the agents gradually reach a stable joint policy. Particularly, the loss functions of DRL agents in our proposed scheduler are finally declined with the training episodes, despite some fluctuations due to random exploration. Besides, different from traditional supervised learning, the DRL kernel is still being updated with training episodes after the loss function tends to be stable.

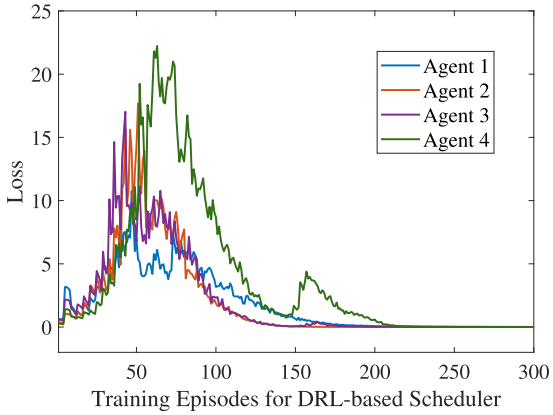


Fig. 9. Training convergence of the proposed multi-agent DRL-based resource scheduler.

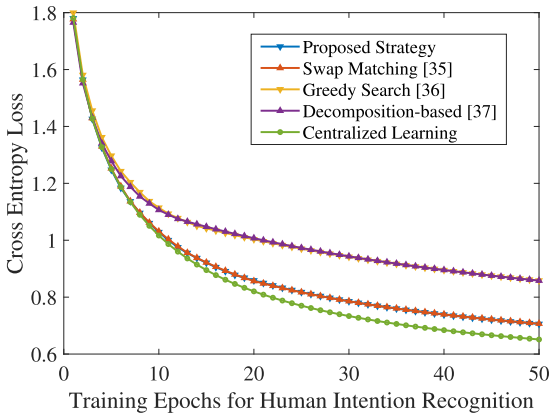


Fig. 10. Loss performance in the learning process of the human intention recognition model.

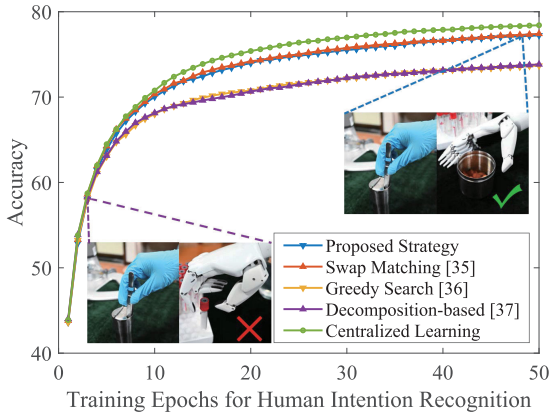


Fig. 11. Accuracy performance in the learning process of the human intention recognition model.

Finally, we evaluate the impact of resource allocation on data utilization in our proposed learning framework. In the following experiments, heterogeneous energy budgets are allocated to clients. Fig. 10 depicts the loss curves of training the intention recognition model under different resource schedulers, where the counterpart of centralized learning without round-by-round feature exchange serves as the performance benchmark. From the given results, we can see that the five loss curves gradually decrease with training epochs. Moreover, the proposed scheduler achieves the similar performance as the swap-matching based method, and outperforms than the other two heuristics. Fig. 11 illustrates the variation of model accuracy achieved by the proposed learning framework with different schedulers and centralized learning. It can be found that the centralized learning achieves the highest model accuracy (at the expense of privacy disclosure). The accuracy curves of the proposed scheduler and swap-matching based baseline are close to that of centralized learning, and are superior to greedy search and subproblem decomposition based ones. The comparison results of adopting different resource schedulers demonstrate the critical role of client association and channel assignment in improving the data utility. The reason for data utility loss of inefficient resource schedulers is that clients with poor energy budgets and worse communication capacities quit in collaborative learning early.

VII. Conclusion

In this work, a privacy-enhanced and spectrum-efficient federated learning scheme in DTMNs was designed. We first proposed a cloud-edge-client collaborative learning framework that amalgamates model splitting and federated aggregation. In particular, the learnable model was divided between clients and servers to provide a higher level of privacy than traditional federated learning. Next, to resolve the training efficiency concerns incurred by frequent feature exchange over wireless networks, we modeled the communication and computation costs of each client in the proposed learning framework, and designed a DT-assisted multi-agent DRL-based resource scheduler for joint client association and channel assignment. Importantly, the emulated network operations in the DT environment were used to train DRL agents, so as to avoid conducting random explorations in real-world systems. Last, experiments on training an intention recognition model for human-robot collaborative nursing demonstrated the cost-saving benefits of the proposed resource scheduler in practical systems.

References

[1] L. Zhou, D. Wu, J. Chen, and X. Wei, "Cross-modal collaborative communications," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 112–117, Apr. 2020.

[2] L. Zhou, D. Wu, X. Wei, and J. Chen, "Cross-modal stream scheduling for eHealth," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 426–437, Feb. 2021.

[3] B. Tan, Y. Qian, H. Lu, D. Hu, Y. Xu, and J. Wu, "Toward a future network architecture for intelligence services: A cyber digital twin-based approach," *IEEE Netw.*, vol. 36, no. 1, pp. 98–104, Jan. 2022.

[4] M. G. Kapteyn, J. V. R. Pretorius, and K. E. Willcox, "A probabilistic graphical model foundation for enabling predictive digital twins at scale," *Nature Comput. Sci.*, vol. 1, no. 5, pp. 337–347, May 2021.

[5] L. Zhao, G. Han, Z. Li, and L. Shu, "Intelligent digital twin-based software-defined vehicular networks," *IEEE Netw.*, vol. 34, no. 5, pp. 178–184, Sep. 2020.

[6] S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox, "Scaling digital twins from the artisanal to the industrial," *Nature Comput. Sci.*, vol. 1, no. 5, pp. 313–320, May 2021.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat.*, Apr. 2017, pp. 1273–1282.

[8] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2019, pp. 14774–14784.

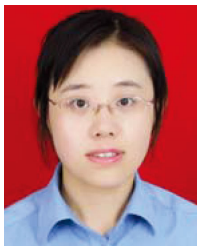
[9] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients—How easy is it to break privacy in federated learning?" 2020, *arXiv:2003.14053*.

[10] X. Jin, P.-Y. Chen, C.-Y. Hsu, C.-M. Yu, and T. Chen, "CAFE: Catastrophic data leakage in vertical federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2021, pp. 994–1006.

[11] A. Falcetta and M. Roveri, "Privacy-preserving deep learning with homomorphic encryption: An introduction," *IEEE Comput. Intell. Mag.*, vol. 17, no. 3, pp. 14–25, Aug. 2022.
[12] K. Wei et al., "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.
[13] K. Wei et al., "User-level privacy-preserving federated learning: Analysis and performance optimization," *IEEE Trans. Mobile Comput.*, vol. 21, no. 9, pp. 3388–3401, Sep. 2022.
[14] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for Internet of Things: Recent advances, taxonomy, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1759–1799, 3rd Quart., 2021.
[15] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5098–5107, Jul. 2021.
[16] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning for digital twin edge networks in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5709–5718, Aug. 2021.
[17] W. Yang, W. Xiang, Y. Yang, and P. Cheng, "Optimizing federated learning with deep reinforcement learning for digital twin empowered industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1884–1893, Feb. 2023.
[18] L. Jiang, H. Zheng, H. Tian, S. Xie, and Y. Zhang, "Cooperative federated learning and model update verification in blockchain-empowered digital twin edge networks," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11154–11167, Jul. 2022.
[19] W. Sun, S. Lei, L. Wang, Z. Liu, and Y. Zhang, "Adaptive federated learning and digital twin for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5605–5614, Aug. 2021.
[20] R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic, "Deep learning for hybrid 5G services in mobile edge computing systems: Learn from a digital twin," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4692–4707, Oct. 2019.
[21] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020.
[22] G. Shen et al., "Deep reinforcement learning for flocking motion of multi-UAV systems: Learn from a digital twin," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11141–11153, Jul. 2022.
[23] F. Tang, Y. Zhou, and N. Kato, "Deep reinforcement learning for dynamic uplink/downlink resource allocation in high mobility 5G HetNet," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2773–2782, Dec. 2020.
[24] M. Akbari, M. R. Abedi, R. Joda, M. Pourghasemian, N. Mokari, and M. Erol-Kantarci, "Age of information aware VNF scheduling in industrial IoT using deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2487–2500, Aug. 2021.
[25] J. Wallenius, J. S. Dyer, P. C. Fishburn, R. E. Steuer, S. Zionts, and K. Deb, "Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead," *Manage. Sci.*, vol. 54, no. 7, pp. 1336–1349, Jul. 2008.
[26] L. Zhang and S. Geng, "The complexity of the 0/1 multi-knapsack problem," *J. Comput. Sci. Technol.*, vol. 1, no. 1, pp. 46–50, Mar. 1986.
[27] J. Foerster et al., "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1146–1155.
[28] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. ICML*, 1993, pp. 330–337.
[29] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[30] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," 2015, *arXiv:1509.06461*.
[31] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," 2015, *arXiv:1511.06581*.
[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[34] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*.
[35] S. Ni, L. Zhao, A. Li, D. Wu, and L. Zhou, "Cross-view human intention recognition for human–robot collaboration," *IEEE Wireless Commun.*, vol. 30, no. 3, pp. 189–195, Jun. 2023.
[36] L. Zhao, D. Wu, L. Zhou, and Y. Qian, "Radio resource allocation for integrated sensing, communication, and computation networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 8675–8687, Oct. 2022.
[37] M. N. H. Nguyen, C. W. Zaw, K. Kim, N. H. Tran, and C. S. Hong, "Let's share the resource when We're co-located: Colocation edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5618–5633, May 2020.
[38] T. LeAnh et al., "Matching theory for distributed user association and resource allocation in cognitive femtocell networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 8413–8428, Sep. 2017.

Lindong Zhao received the Ph.D. degree in signal and information processing from the Nanjing University of Posts and Telecommunications, China, in 2023. He is currently a Lecturer with the School of Communications and Information Engineering, Nanjing University of Posts and Telecommunications. His research interests include multimedia communications and computing.

Shouxiang Ni (Graduate Student Member, IEEE) received the B.E. degree in communication engineering from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2021, where he is currently pursuing the Ph.D. degree in information and communication engineering. His research interests include multimedia communications.



Dan Wu received the Ph.D. degree in electronic engineering from the PLA University of Science and Technology, China, in 2012. She is currently a Professor with the Army Engineering University of PLA, China. Her research interests include multimedia communications and wireless communications.

Liang Zhou (Senior Member, IEEE) received the joint Ph.D. degree in electronic engineering from École Normale Supérieure (ENS), Cachan, France, and Shanghai Jiao Tong University, Shanghai, China, in 2009. He is currently a Professor with the Nanjing University of Posts and Telecommunications, China. His research interests include multimedia communications and computing.