# Joint Sensing Adaptation and Model Placement in 6G Fabric Computing

Yixue Hao, *Member, IEEE*, Long Hu, and Min Chen, *Fellow, IEEE*

***Abstract*— Sensing and computing based on intelligent fabrics can meet the ultra-reliable and low-latency communication (URLLC) needs of sixth-generation wireless (6G) by integrating sensing units into fabric fibers to perceive user data. Although some researchers have designed sensing or computing solutions, such solutions have not been well explored. In this paper, we consider the joint sensing adaptation and model placement in a 6G fabric space. We first propose an intelligent-fiber-driven 6G fabric computing network to minimize acquisition latency while ensuring accuracy. Then, we formulate an optimization model that takes the fabric sampling rate, sampling density, and model placement as variables. To solve the model, we propose an effective learning algorithm based on deep reinforcement learning. That is, by transforming the optimization problem into a state space, action space, and reward function, we design an optimal sensing and placement scheme. The simulation results show that our proposed scheme can achieve optimal sensing and computing compared with several baseline algorithms.**

***Index Terms*— 6G network, intelligent sensing, model placement.**

## I. Introduction

With the advancement of communication technology, fifth-generation wireless (5G) network technology has been widely used in Internet of Thing (IoT) sensing (e.g., industrial scenarios) [1], [2]. In these scenarios, traditional sensing refers to devices that utilize high bandwidth to transmit the sensed large amount of data to the cloud or edge cloud for processing [3]. However, these sensing devices have a certain size and shape, and owing to the user's mobility, the devices lack of long-term interactions between the sensing device and the user, making it difficult to achieve senseless sensing in the user's living environment. However, one goal of sixth-generation wireless (6G) networks is to realize ultra-reliable and low-latency communication and enable ubiquitous sensing and intelligence for users. Thus, it is important to design new sensing technology to enhance the 6G network [4].

Manuscript received 22 October 2022; revised 31 January 2023; accepted 7 March 2023. Date of publication 5 June 2023; date of current version 6 July 2023. This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 62276109, Grant 62176101, and Grant 61821003. The work of Yixue Hao was supported by the Hubei Natural Science Foundation under Grant 2021CFB050. *(Corresponding author: Min Chen.)*

Yixue Hao and Long Hu are with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: yixuehao@hust.edu.cn; hulong@hust.edu.cn).

Min Chen is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510641, China (e-mail: minchen@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3280968.

Digital Object Identifier 10.1109/JSAC.2023.3280968

Fortunately, by fusing functional fibers with sensing units to form intelligent fabrics [5], [6], the fabric computing is proposed [7]. Moreover, we can sense a large amount of data about the user, and the space composed of intelligent fabrics is called the fabric space. In the fabric space, the device is hidden and has sensing and computing capacity [8]. Multidimensional data about the user can be sensed in a non-intrusive manner, which helps to provide services to the user. Thus, by deploying intelligent fabrics in the user's living space, senseless sensing can be achieved.

However, compared to traditional sensing devices, fabric sensors are characterized by the ultra-dense deployment of sensing nodes and long monitoring times, resulting in the temporal correlation of fabric sensing data. If all the ultra-dense sensing correlation data are transmitted for analysis, a long delay can occur, making a large amount of sensing data outdated. Thus, a reasonable sensing sampling rate should be designed to achieve high-quality data sensing.

Moreover, the data sensed by the fabric have an inconsistent quality density. For example, when a user presses a fabric sensing unit, a higher pressure is detected at the press point, while other neighboring units detect a lower pressure. Thus, we need to control the sampling density of sensing data. Based on the reasonable control of sampling rate and density of sensing data, we can achieve the high-quality transmission of data and reduce the latency [9]. Moreover, for sensing data, ensuring the intelligence of the fabric space is also a challenge.

To address the above challenges, we should deploy the artificial intelligent (AI) computing model (i.e., a deep neural network or DNN model) at the edge cloud that is closed to the fabric space. Implementing intelligence at the edge is a feasible solution that has attracted considerable attention [10], [11]. For example, by deploying DNN models close to the data side, researchers can achieve high-accuracy recognition using transfer learning [12], [13]. However, the recognition capability of the deployed models is limited by the perceived data and the size of the models, and the inference delay varies between different DNN models.

Based on the above analysis, two challenges exist for sensing and computing in the 6G fabric computing network. The first challenge is complexity of latency-accuracy tradeoff. If we deploy more complex computational models in the fabric space, using a higher sampling rate and density, we will achieve higher recognition accuracy but a longer transmission delay. The second challenge is that there is a complex relationship among fabric sampling rate, sampling density, and DNN

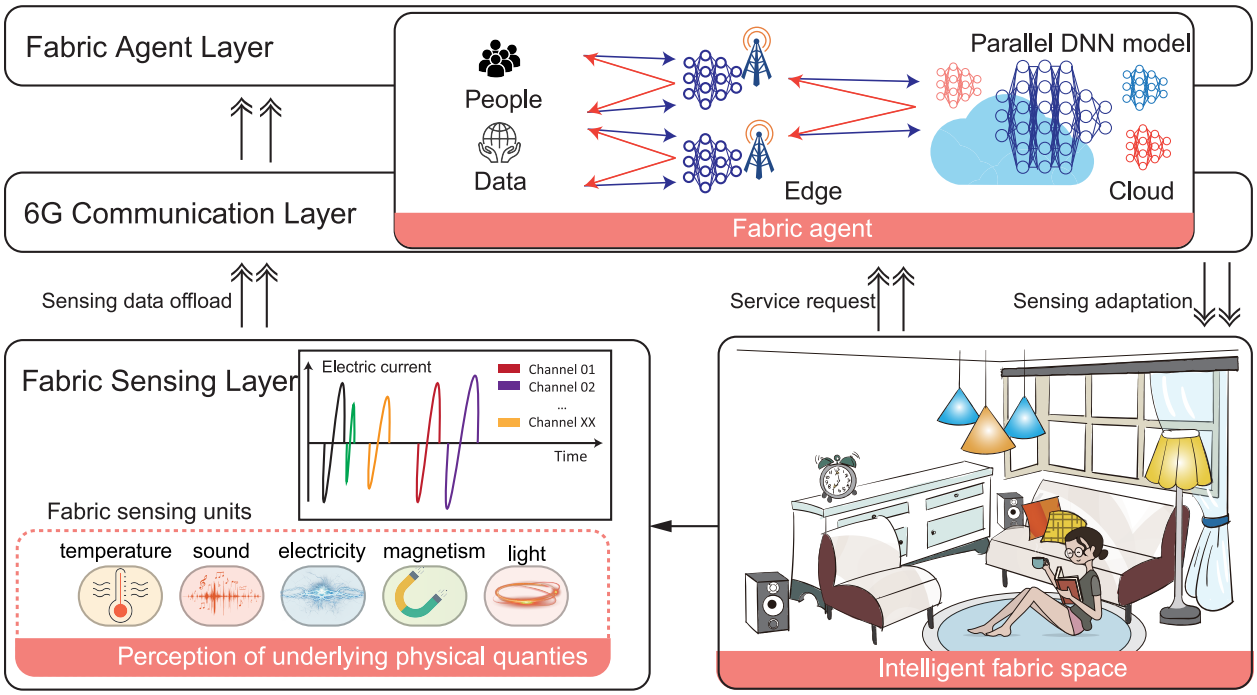


Fig. 1. An illustration of architecture of 6G fabric computing network.

model, and they have different impacts on system delay and accuracy. Thus, our work considers joint sensing adaptation and model placement in the fabric space.

To address these challenges, we propose an Intelligent Adaptive Sensing and Computing (IASC) scheme. Specifically, we first propose an intelligent fabric agent-based sensing and computing framework. Then, we formulate an optimization model with model placement, sampling rate, and density control. To solve this model, we use deep reinforcement learning (DRL) to achieve the optimal model placement and optimal control of the sampling rate and density. Finally, we experimentally verify the effectiveness of the IASC algorithm.

In summary, the main contributions of this paper are list as follows:

- Fabric agent-based sensing and computing framework: Based on the fabric computing, edge computing and 6G network, we first propose an intelligent sensing and computing architecture driven by functional fibers, which includes a fabric sensing and computing layer, URLLC layer, and fabric agent layer.
- IASC scheme: We formulate a delay minimization and accuracy maximization problem for 6G fabric computing which is nonlinear programming problem. To address this problem, based on DRL, we propose an effective online sensing and computing scheme, the IASC scheme. This algorithm realizes the intelligent online control of the sampling rate and sampling density of fabric sensing data, and it can achieve optimal model placement.
- Extensive performance evaluation: To evaluate the IASC scheme, we conduct extensive experiments. The experimental results show that compared with various baseline algorithms, the IASC algorithm can be adaptive, minimize the sensing and computing delay, and ensure high accuracy. Moreover, we analyze the effect of the number of fabrics on the algorithm.

The remainder of this paper is organized as follows. Section II reviews related work. The system model and problem are described in Section III. We present the IASC schemes in Section IV. Our simulation results and discussion are given in Section V. Finally, Section VI provides the conclusion of this paper.

## II. Related Works

5G establishes a new framework for wirelessly connected core network communications [14] and can connect more sensing devices at higher speeds, facilitating intelligent connectivity, sensing, and communication for the IoT. In the 5G sensing network, multimodal physical parameters are collected from humans, the environment, and many machines using sensors. Then, using 5G communication technology, the sensors transmit the data to the cloud and processes it by deep learning techniques, which can enable pattern recognition, optimal control, and other intelligent services [15]. The two main features are as follows:

- Spatially distributed sensing networks: A 5G sensing network is a computer network of spatially distributed automated devices that collaboratively monitor physical or environmental conditions at different locations through sensors and collect relevant information.
- High-bandwidth and low-latency communication: 5G equips the IoT with low latency, high speed, and multi-terminal interaction capabilities. IoT devices are capable of responsive human-computer interaction with users, thus providing more frequent, intelligent, and automatic service modes.

Thus, 5G mainly highlights intelligent applications in the industry, which enables almost intelligent sensing. However,

the sensing devices are difficult to deploy in users' living environments and users resist them owing to the foreign and controlling nature of sensing devices. In the 6G network, human-centric sensing and communications are emphasized. Therefore, 6G sensing technology has more requirements for user quality of service and supports heterogeneous sensing and intelligence [16], [17]. Thus, the 6G network more realistically approaches the physical location where motion occurs, enables a more accurate capture of physical physiological indicators, and performs interactive actions more similarly to the human body.

Fortunately, multifunctional fibers can measure multimodal raw physical quantities such as sound, light, force, heat, electricity, and magnetism, and they can also provide feedback on human interaction by changing color [18] and regulating temperature [19]. The delicate, soft fabric can be seamlessly embedded in the living environment to capture the first-hand data of the user's contact with the physical world without disturbance, which helps to seamlessly map our physical world to the digital space. Therefore, a new computing paradigm for fabric computing is proposed. Moreover, we can use fabric computing to sense physiological indicators, activity behavior, and other information and realize self-management, self-configuration, and self-optimization [20].

Compared to 5G IoT sensing, 6G fabric computing collects data in a field closer to the user to achieve a high level of comfort for sensing and interaction. Moreover, fabric computing is covert and can bring more freedom to users. At the same time, the user can coexist with the terminal device for a long time in a non-disturbing and non-sensitive sensing environment, collecting user data in all aspects. These data with spatiotemporal characteristics are fed back to the learning system to provide highly intelligent services. Thus, compared to traditional concept of an intelligent sensing based on pervasive computing [21], [22] the hidden nature of the fabric sensing drives causes them to be unnoticeable in the living space, which is more practical.

In addition, compared with traditional communication networks, 6G networks are not only basic communication facilities but also deeply integrated AI models from multiple dimensions, such as connectivity, computing power, algorithms, and data. Thus, the networks should build true intelligence. The intelligent controlling algorithm can be applied to communication optimization to improve network performance, efficiency, and user service experience and build resource management optimization to support mobile or distributed scenarios [23].

## III. System Model and Problem Description

In this section, we first present a 6G fabric computing network. Then, by analyzing the communication latency, computing latency, and model accuracy, we establish an optimization problem with the goal of minimizing latency and maximizing accuracy.

### A. System Architecture

The 6G fabric computing network uses fabric sensing units to capture user data at multiple spatial and temporal scales. The large amount of heterogeneous data is transmitted to the fabric agent layer through the 6G network with low latency and high reliability. The 6G fabric computing network can meet users' intelligent service needs, with fabric computing as the center and the 6G communication network as the basis for end-to-end, interference-free intelligent sensing. As shown in Fig.1, the system consists of three layers:

- Fabric sensing layer: A large number of multifunctional fibers are used as multichannel, multimodal sensing units, and can measure multimodal raw physical quantities such as temperature, sound.
- 6G communication layer: These fabric data carry information such as the location and temperature transmit to the edge cloud or cloud through 6G communication. The 6G network is deployed to realize personalized services for users by sensing their behavior and intentions. When user requirements change, the fabric computing network can seamlessly switch service methods and communication policies.
- Fabric agent layer: The fabric agent deployed in the edge cloud and processed the data obtained in the fabric space which are multidimensional, with temporal, spatial, and intensity characteristics. And it performs a reliable behavior and intention pattern analysis, and achieve all-around cognition.

Fig. 2 shows the fabric agent in the 6G fabric computing network. The purpose of the fabric agent is to realize non-disturbed intelligent perception and computing based on the limited communication, computing, and storage resources in fabrics units. In this system, we need to solve the following problems: 1) Model placement. This refers to how to deploy an intelligent computing model for fabric data to achieve accurate sensing of the user's request; 2) Sensing adaptation. This refers to how to control the sampling rate and density of the fabric through the fabric agents and then affect the transmission latency, computation latency, and accuracy.

### B. DNN Placement Model

We consider a fabric space in 6G environment that consist of $N$ fabrics, $\mathcal{N} = \{1, 2, \cdots, i, \cdots, N\}$, which connect to the fabric agent in edge cloud through URLLC mode. For these fabrics, the collected data are offloaded to the fabric agent for calculation and processing, such as a user behavior analysis. We assumed that there are $M$ intelligent DNN learning models that can analyze the offloaded data. The intelligent learning model is recorded as $\mathcal{M} = \{1, 2, \cdots, j, \cdots, M\}$ which can input data collected from fabrics with different sampling rate and density.

Based on the above discussion, we define the variable of the fabric $i$ selection the model $j$ (i.e., model $j$ is placed on the fabric agent) as $x_{i,j}$, which can be expressed as follows:

$$x_{i,j} = \begin{cases} 1 & \text{The fabric } i \text{ select model } j, \\ 0 & \text{The fabric } i \text{ does not select the model } j. \end{cases} \tag{1}$$

In this work, we assume that if a DNN computation model $j$ is placed on the fabric agent, but the sampling rate and

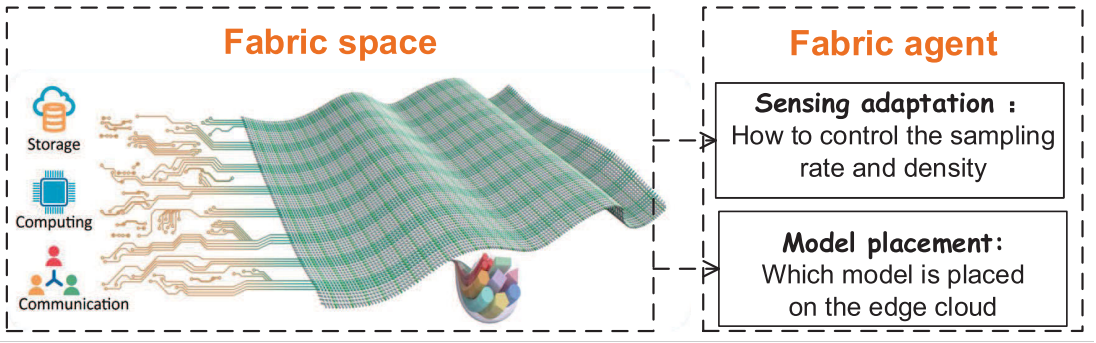


Fig. 2. Fabric agent in the 6G fabric computing network.

density of the fabric $i$ are not fixed; that is, the model $j$ can process data with different sampling rates and densities. And the sampling rates and densities are determined by the fabric agent. Moreover, we use $c_j$ to denote the size of the DNN model $j$. Considering the limited capacity of edge cloud storage, we define the total capacity of fabric agent for model placement as $C$, we can obtain the constraint as follows:

$$\sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j}c_j \leq C. \tag{2}$$

### C. Service Latency Model

The amount of data transmission for fabric is determined by the sampling rate and sampling density. Assume that there are $K$ types of sampling rates available. Therefore, we can define the set of fabric sampling rates as $r_{sa} = \{r_{sa}^1, r_{sa}^2, \cdots, r_{sa}^k, \cdots, r_{sa}^K\}$. For the simplify, we assume that $r_{sa}^1 < r_{sa}^2 < \cdots < r_{sa}^K$, and when $r_{sa}^k$ is smaller, less data are collected. Otherwise, more data are sensing. Moreover, we define the decision variable of the sampling rate for the fabric $i$ as $y_{i,k}$, where

$$y_{i,k} \in \{0, 1\}, \tag{3}$$

and when $y_{i,k} = 1$ means that the sampling rate of fabric $i$ is $r_{sa}^k$.

Furthermore, considering that the fabric is woven from fabric fibers, for example, assuming that the fabric is woven from $96 \times 96$ fabric electrodes, the sampling density can be $96 \times 96$, $48 \times 48$, $32 \times 32$ or others. Thus, we can assume there are $S$ densities that can be sampled, and we can define the set of fabric sampling density as $D = \{d_1, d_2, \cdots, d_s \cdots, d_S\}$. Moreover, we assume that $d_1 < d_2 < \cdots < d_s$. Furthermore, we define the sampling density variable of the fabric $i$ as $z_{i,s}$, where

$$z_{i,s} \in \{0, 1\}. \tag{4}$$

when $z_{i,s} = 1$, the sampling density of fabric $i$ is $d_s$.

In this work, we assume that the sampling rate and density does not change again after $y_{i,k}$ and $z_{i,s}$ are determined by the fabric agent. Based on this, we can define the amount of data transmission by the fabric $i$ to the fabric agent in edge cloud as $\delta_i$, which can be expressed as follows

$$\delta_i(y_{i,k}, z_{i,s}) = \phi_i(y_{i,k}, z_{i,s})\sigma \tag{5}$$

where $\sigma$ denotes the data size per unit density, and the $\phi_i(\cdot)$ function denotes the number of densities transmitted in one second based on the sampling rate and density, which is calculated as follows:

$$\phi_i(y_{i,k}, z_{i,s}) = \left(\sum_{k=1}^{K} y_{i,k}r_{sa}^k\right)\left(\sum_{s=1}^{S} z_{i,s}d_s\right) \tag{6}$$

where $\sum_{k=1}^{K} y_{i,k}r_{sa}^k$ denotes the sampling rate selected for the fabric $i$, and $\sum_{s=1}^{S} z_{i,s}d_s$ denotes the density of data in the fabric $i$ in one time.

The fabric data transmit to the fabric agent in edge cloud through the URLLC mode and let $r_i$ be the transmission rate of fabric $i$. Specifically, based on the previous design scenario, there are multi-antennas are configured on the edge cloud for $\mathcal{N}$ single-antenna fabric, and there are $\mathcal{P}$ uniformly randomly distributed static scatters. Assume that the transmitter of the fabric transmits at frequency $f$ (wavelength $\lambda_c$). Thus, the channel coefficient between the transmitter and receiver of fabric $i$ can be represented as follows:

$$h_i = \frac{1}{\sqrt{\mathcal{P}}}\sum_{i=1}^{\mathcal{P}} \exp\left(j\frac{2\pi(d_i^{Rx} + d_i^{Tx})}{\lambda_c}\right), \tag{7}$$

where $d_i^{Rx}$ is the distance from the scatterer $i$ to the receiver, and $d_i^{Tx}$ is the distance from the scatterer $i$ to the transmitter. The normalization of $\frac{1}{\sqrt{\mathcal{P}}}$ keeps the same marginal variance under different numbers of scatterers. Let $\mathcal{H}_i = \sqrt{g_i}h_i$, where $h_i$ is the small-scale fading follow the complex Gaussian distribution $\mathcal{CN}(0, I)$. According to Shannon's formula, it can be known that the signal-to-noise ratio, bandwidth and noise have a great influence on the channel capacity. Thus, we can calculates the signal-to-noise ratio as follows

$$\gamma(b_i, p_i) = \frac{p_i||\mathcal{H}_i||}{b_i B N_0}, \tag{8}$$

where $p_i$ and $\mathcal{H}_i$ is the transmit power and the channel coefficient of the fabric $i$, respectively. And $b_i$ is the bandwidth allocated on the fabric $i$, $B$ is the system bandwidth, $N_0$ is unilateral noise, then the uplink URLLC transmission rate (bit/s) is:

$$r_i \approx \frac{B}{\ln 2}[b_i \ln(1 + \gamma(b_i, p_i)) - \sqrt{\frac{b_i V_i(b_i, p_i)}{\varphi B}}Q^{-1}(\epsilon_i)]. \tag{9}$$

where $\varphi$ is the transmission time interval, $\epsilon_i$ is the decoding error probability, $\gamma(b_i, p_i)$ is the signal-to-noise ratio of the link channel between the fabric $i$ and the edge cloud, $Q^{-1}(\cdot)$ is the inverse function of the standard Q function, and $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} \exp(\frac{-t^2}{2})dt$, $V_i$ is the channel dispersion. According to [24], under the complex Gaussian channel with limited average power, $V_i$ can be calculated as follows:

$$V_i = \gamma \frac{2+\gamma}{(1+\gamma)^2} \frac{1}{(\ln 2)^2}. \tag{10}$$

so we can obtain the data transmission delay from fabric $i$ to fabric agent in edge cloud as

$$T_i^{tra}(y_{i,k}, z_{i,s}) = \frac{\delta_i(y_{i,k}, z_{i,s})}{r_i}. \tag{11}$$

Now, we give the fabric data computation latency. Specifically, the computation delay is determined by the sampling rate, sampling density, and the DNN model placement in edge cloud. For example, when the sampling rate and density are low, the amount of data processed by the DNN model is also small, so its computation delay is small. Thus, when given the DNN placement strategy $x_{i,j}$, sampling frequency $y_{i,k}$, and sampling density $z_{i,s}$, the fabric $i$ required computation resource $v_i$ in fabric agent can be obtained as

$$v_i(x_{i,j}, y_{i,k}, z_{i,s}) = \sum_{j=1}^{M} x_{i,j} \eta_j \delta_i(y_{i,k}, z_{i,s}), \tag{12}$$

where $\eta_j$ denotes the computational intensity of the DNN model $j$, that is, the processor cycles to be consumed per bit of data. Generally speaking, the larger the model, the greater the amount of computation required. We can obtain the processing delay of the fabric $i$ as follows:

$$T_i^{comp}(x_{i,j}, y_{i,k}, z_{i,s}) = \frac{v_i(x_j, y_{i,k}, z_{i,s})}{f_i^c} \tag{13}$$

where $f_i^c$ denotes the computation frequency (cycles/s) of the fabric agent in the edge cloud allocated to the fabric $i$. Considering that delay is an important metric to measure the quality of user experience, we can define the total delay of fabric $i$ as follows:

$$T_i = T_i^{tra}(y_{i,k}, z_{i,s}) + T_i^{comp}(x_{i,j}, y_{i,k}, z_{i,s}) \tag{14}$$

### D. Analysis Accuracy Model

In addition to considering the latency of fabric data sensing, we need determine the impact of the collected fabric data on the accuracy of the decision user's request. We assume that when the fabric sensing data sampling rate $y_{i,k}$ and sampling density $z_{i,s}$ are fixed as $y_{i,k^*}$ and $z_{i,s^*}$, respectively, the model baseline accuracy is

$$\mathcal{A}_{base} = \{a_1^{k^*,s^*}, a_2^{k^*,s^*}, \cdots, a_j^{k^*,s^*}, \cdots a_M^{k^*,s^*}\}, \tag{15}$$

where $a_j^{k^*,s^*}$ represent the accuracy of model $j$ when the sampling rate is $r_{sa}^{k^*}$ and the sampling density is $d_{s^*}$, and $a_j^{k^*,s^*} \in [0,1]$. Existing research has found that the sampling rate and density of fabrics independently affect the accuracy of the model, and the larger the size and computational complexity of the model, the higher the accuracy of the model. Thus, when the sampling rate and sampling density are $y_{i,k}$ and $z_{i,s}$, compared to the model baseline accuracy, the the effect on the accuracy is expressed using the functions $p_j(\cdot)$, so that the accuracy of the fabric $i$ can be defined as follows:

$$A_i(x_{i,j}, y_{i,k}, z_{i,s}) = \sum_{j=1}^{M} x_{i,j} p_j(y_{i,k}, h_{i,s}) a_j^{k^*,s^*}. \tag{16}$$

where $p_j \in [0,1]$, and it can be obtained by fitting different models in historical data with different sampling rates and densities of the fabric data.

### E. Problem Formulation

In this work, we focus on two problems: 1) the fabric sensing problem, that is, how to set the sampling rate and sampling density of fabric sensors to be able to achieve low latency data sensing; 2) the DNN model placement problem, that is, how to deploy DNN models on the fabric agent in the edge cloud to achieve an accurate decision. For better representation, we can define the gain of the fabric $i$ as follows:

$$U_i(x_{i,j}, y_{i,k}, z_{i,s}) = -\omega_1 T_i + \omega_2 A_i \tag{17}$$

where $\omega_1$ and $\omega_2$ are weighting parameters for latency and accuracy, respectively. Thus, our goal is to minimize the latency and maximum accuracy of perception and computation (i.e., the gain of fabric sensing and computation), which can be formulated as follows:

$$\begin{aligned}
\mathcal{P}: \max_{\{x_{i,j}, y_{i,k}, z_{i,s}\}} \ & \frac{1}{N} \sum_{i=1}^{N} U_i(x_{i,j}, y_{i,k}, z_{i,s}) \\
s.t. \quad & \mathcal{C}1: x_{i,j} \in \{0,1\} \quad i \in \mathcal{N}, j \in \mathcal{M} \\
& \mathcal{C}2: \sum_{i=1}^{N} \sum_{j=1}^{M} x_{i,j} c_j \le C. \quad i \in \mathcal{N}, j \in \mathcal{M} \\
& \mathcal{C}3: y_{i,k} \in \{0,1\}, \quad i \in \mathcal{N}, k = \{1,2,\cdots,K\} \\
& \mathcal{C}4: \sum_{k=1}^{K} y_{i,k} = 1 \quad i \in \mathcal{N}, k = \{1,2,\cdots,K\} \\
& \mathcal{C}5: z_{i,s} \in \{0,1\} \quad i \in \mathcal{N}, s = \{1,2,\cdots,S\} \\
& \mathcal{C}6: \sum_{s=1}^{S} z_{i,s} = 1 \quad i \in \mathcal{N}, s = \{1,2,\cdots,S\}
\end{aligned}$$

where constraint $\mathcal{C}1$ indicates that the value of the model variable placed on the fabric agent is 0 or 1. Constraint $\mathcal{C}2$ ensure that the the placement model cannot exceed the capacity of the edge cloud. Constraint $\mathcal{C}3$ and $\mathcal{C}5$ are the range of values for the fabric sampling rate and density variable, respectively. Finally, constraint $\mathcal{C}4$ and $\mathcal{C}6$ mean one and only one sampling rate and density can be selected by the fabric $i$. This problem is a nonlinear programming problem which is NP-hard.

## IV. Intelligent Adaptive Sensing and Computing Scheme

In this section, we give the the intelligent adaptive sensing and computing (IASC) scheme using deep reinforcement learning.

### A. Problem Transformation

In order to solve the optimization problem $\mathcal{P}1$, the problem is first redefined using the Markov Decision Process (MDP). The implementation of the IASC algorithm relies on three elements that are closely related to the environment: state space, action space and reward. Specifically, DRL is performed by an agent observing the current state of the environment, selecting the appropriate action to be executed by the environment, and calculating the corresponding reward to be returned to the agent, which repeats the process so that the algorithm can obtain the maximum reward from the environment after giving the action based on the current state. The state space, action space, and reward defined at time slot t are as follows:

*1) State Space:* In this work, in order to give a better design of the state space, considering that the model placement of the fabric agent determine the sampling rate and density, thus, from the perspective of the intelligent model placement, we task the computational intensity $\eta_j^t$, the transmission rate $r_j^t$ when model $j$ is placed on edge cloud, model accuracy $A_j^t$ and size $c_j^t$ as the system state, which as follows:

$$s_t = \{\eta_1^t, r_1^t, A_1^t, c_1^t; \cdots, \eta_M^t, r_M^t, A_M^t, c_M^t\} \tag{18}$$

*2) Action Space:* The action space contains three variables: DNN model placement decision $x_{i,j}^t$, fabric sampling rate $y_{i,k}^t$ and sampling density $z_{i,s}^t$. To optimize the action space, the sampling rate and density of fabric $i$ at time slot $t$ are set to $r_{sa,t}^k$ and $d_s^t$, respectively. For the fabric space, considering that the variables $x_{i,j}^t$ are discrete, there are $N^M$ choices of deployment models for just deploying fabric data, which leads to an excessive search space with the increasing number of fabrics and models.

Thus, in order to simplify the action space, the action space is redefined. Similar to the definition of the potential energy function, a continuous variable $g_j^t$ ranging from 0 to 1 is defined as the expected gain (i.e., reduction in latency and improvement in accuracy) of DNN model $j$ at time slot $t$, which as follows:

$$g_j^t \in (0,\ 1], \quad j = 1, \cdots, M \tag{19}$$

where $g_j^t$ is larger, the higher the expected gain of the DNN model $j$ at time slot $t$, meaning the higher the probability of being chosen. In this work, the fabric agent selects the DNN model $j$ from the highest to the lowest expected return. Thus, we can define the action space as follows:

$$a_t = \left\{r_{sa,t}^1, d_1^t, g_1^t; r_{sa,t}^2, d_2^t, g_2^t; \cdots, r_{sa,t}^M, d_M^t, g_M^t\right\} \tag{20}$$

Furthermore, considering the diversity of user requirements, we assume that $q_j$ DNN models are placed at one time slot, and $\sum_{j=1}^{M} q_j \leq C$. Thus, the action space has changed to selecting $q_j$ DNN model from the original space from large to small, which is noted as $a_t^*$. Then the decision of the DNN service model, fabric sensing data sampling rate and density at time slot $t$ can be defined as

$$a_t^* = \{a_{t,1}^*, a_{t,2}^*, \cdots, a_{t,q_j}^*\}, \tag{21}$$

Furthermore, in order to improve convergence speed, we range the action space to $(-1, 1)$, the scaling function as follows:

$$a_t = \begin{cases} r_{sa,t}^j = r_{sa}^{\min} + \dfrac{1}{2}\left(r_{sa}^{\max} - r_{sa}^{\min}\right)\left(r_{sa,j}^t + 1\right), \\ d_j^t = d_{\min} + \dfrac{1}{2}\left(d_{\max} - d_{\min}\right)\left(d_j^t + 1\right), \\ g_j^t = \dfrac{1}{2}\left(g_j^t + 1\right). \end{cases} \tag{22}$$

Through the above analysis, we build the action space.

*3) Reward:* The reward function is the feedback information given by the environment to the fabric agent in the edge cloud, and the fabric agent decides whether the action is good or bad based on the reward function. Our aim is to minimize the $U_i^t(x_{i,j}, y_{i,k}, z_{i,s})$, so the reward function can be defined as

$$r_t = \sum_{i=1}^{N} U_i^t(x_{i,j}, y_{i,k}, z_{i,s}). \tag{23}$$

The fabric agent searches for larger $r_t$ during the learning process, so that $U_i^t(x_{i,j}, y_{i,k}, z_{i,s})$ becomes larger, resulting in lower delay and higher accuracy.

### B. The IASC Algorithm

Based on the above introduction, we transform the problem into an MDP tuple. To achieve the solution of the problem, we design the actor-critic networks. Specifically, actor is the policy network $\Pi_{\theta^\mu}$ which is responsible for outputting actions and interacting with environment. The input of $\Pi_{\theta^\mu}$ is the state $s_t$ at time slot $t$, and the output is the policy $\pi(s_t)$ at time slot $t$. The critic is the Q-network, which evaluates the action output at each step and estimates the future reward $Q$ value of the action to guide the actor network to generate a better action in the next step, i.e., $Q_{\theta^Q}(s_t, a_t)$, where $s_t$ is the state space, $a_t$ is the action space, and $\theta^Q$ is the weight parameter of the Q-network. Thus, the input of the critic network is the joint vector of action and state, and the output of the critic is the $Q$ value. If $r(s_t, a_t)$ means the reward in the state $s_t$ and executing action $a_t$ at time slot $t$, the $Q$ value function can be expressed as

$$Q_{\theta^Q}(s_t, a_t) = E[r(s_t, a_t) + \gamma Q_{\theta^Q}(s_{t+1}, \pi_{\theta^\mu}(s_{t+1})] \tag{24}$$

where $\gamma$ is the discount factor used to determine the effect of Q value after N steps.

To update critic networks, actor network is used to interact with the environment to generate rewards, which is the part of TD-error. The other part of TD-error is output by the target critic network $Q_{target}$, and its parameters are updated by critic in accordance with a certain frequency, in order to make the training more stable. TD-error or the loss function of critic network can be defined as

$$\delta_{loss} = r(s_t, a_t) + \gamma Q_{target}\left(s_{t+1}, \pi(s_{t+1})\right) - Q(s_t, a_t), \tag{25}$$

when TD-error is computed, we can minimize $\delta_{loss}$ using mean squared error method to update critic network. However, for the update of actor networks, we utilize the method of

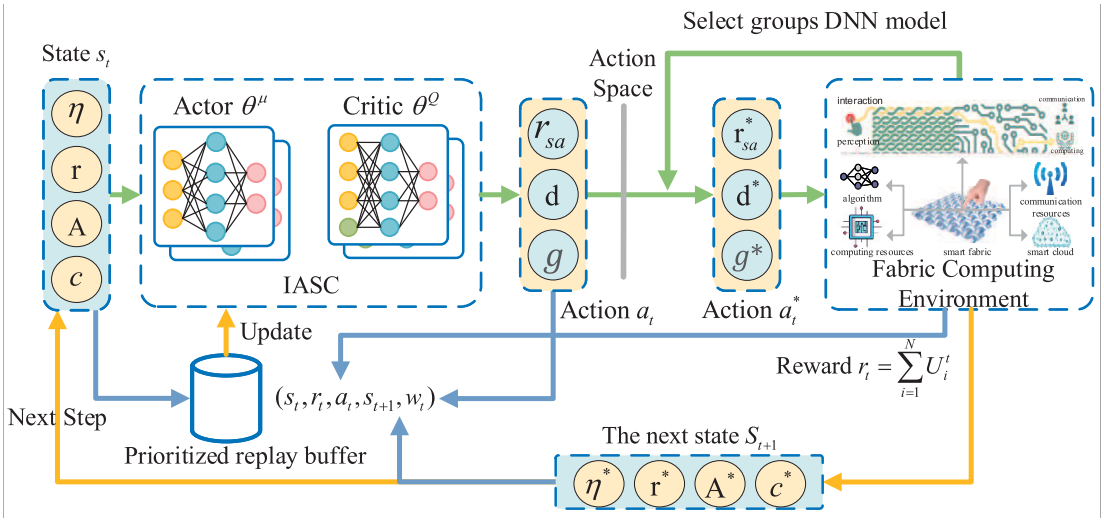


Fig. 3. The intelligent adaptive sensing and computing (IASC) algorithm.

gradient ascent algorithm. Thus, with the mini-batch size $m_{ms}$, the loss function of actor networks can be expressed as follows:

$$L_A = -\frac{1}{m_{ms}} \sum_{i=1}^{m_{ms}} Q(s_t, a_t, \theta^Q) \tag{26}$$

The IASC algorithm as shown in the Fig. 3, which includes state space, action space, reward function, and fabric environment. The details of the algorithm are described below.

In this work, we use the same method as deep deterministic policy gradient (DDPG) [25] to give the action that maximizes the value of Q, which can make the reward maximum. However, Markovianity exists in the data for reinforcement learning, which does not satisfy the prerequisite assumption that samples need to be independent and identically distributed for training neural networks, and the training process is very unstable when using neural networks for reinforcement learning. Borrowing from the DQN algorithm using experience replay, the target networks were built separately for the Q network and the policy network during the neural network training. We define the parameters of the actor online policy network as $\theta^{\mu}$, the parameters of the actor target policy network as $\theta^{\mu^*}$, the parameters of the critic online Q network as $\theta^{Q}$, the parameters of the critic target Q network as $\theta^{Q^*}$, and the loss function as $J$. We can obtain the update formula for the actor network as

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \nabla_a Q\left(s, a \middle| \theta^Q\right) |_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu\left(s \middle| \theta^{\mu}\right) |_{s_i} \tag{27}$$

where $\nabla_a Q\left(s, a \middle| \theta^Q\right)$ is the direction of the maximum value function passed from the critic network. Furthermore, we can obtain the update formula for the critic network as follows:

$$\begin{aligned} y_i &= r_i + \gamma Q^*\left(s', \mu^*\left(s' \middle| \theta^{\mu^*}\right) \middle| \theta^{Q^*}\right), \\ L &= \frac{1}{N} \sum_i \left(y_i - Q\left(s_i, a_i \middle| \theta^Q\right)\right)^2 \end{aligned} \tag{28}$$

where $y_i$ is the estimated value of the target in reinforcement learning, and $(s_i, a_i, r_i, s_i')$ is the state, action, reward value and next action sampled from the experience pool. The target network parameters are soft updated as follows:

$$\begin{cases} \theta^{Q^*} = \tau\theta^Q + (1-\tau)\,\theta^{Q^*}, \\ \theta^{\mu^*} = \tau\theta^{\mu} + (1-\tau)\,\theta^{\mu^*}, \end{cases} \tag{29}$$

where $\tau$ denotes the soft update weight, which can prevent a certain update from being too large to improve the stability of the algorithm learning.

In fact, with the increase of the number of fabrics and models, we find that the time required for training increases sharply and the reward is sparse. Therefore, we improve the original replay buffer to the prioritized replay buffer to reduce the time required for convergence and alleviate the reward sparsity problem. Specifically, unlike the original experience replay, which is randomly sampled from the experience pool, the prioritized experience replay uses the TD-error method to calculate the priority. The larger the priority, the greater the probability of being sampled. The larger TD-error represents the greater value of the experience, and its sampling probability will be improved. However, the whole training process does not completely sample the experience with the large priority. The data with a small TD-error also be sampled with a small probability, otherwise overfitting will occur. The prioritized experience replay exploits a data structure called sumtree to store experience and priorities, and utilizes importance sampling to calculate weights to correct sampling distribution errors caused by priority sampling. We assume that the number of leaf nodes of the sumtree is $st$, the priority of each leaf node is $pr_i$, the important sampling weight is $sw_j$, and the sampling weight coefficient is $\beta$. Based on this, the updating formula of priority can be defined as

$$TD_i = y_i - Q\left(s_i, a_i \middle| \theta^Q\right), pr_i = |TD_i| \tag{30}$$

and the importance sampling weights $sw_i$ can be expressed as follows:

$$\begin{aligned} sw_i &= \frac{(NPr(i))^{-\beta}}{\max_i (sw_i)} \\ &= \frac{(NPr(i))^{-\beta}}{\max_i (NPr(i))^{-\beta}} \\ &= \left(\frac{Pr(i)}{\max_i Pr(i)}\right)^{-\beta} \end{aligned} \tag{31}$$

where $Pr(i)$ is the probability of being sampled, that is

$$Pr(i) = \frac{pr_i}{\sum_k (pr_k)} \tag{32}$$

so, the update formula for the critic network is modified to

$$L = \frac{1}{N}\sum_i sw_i \left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)^2 \tag{33}$$

In summary, the complete algorithm is presented in Algorithm 1. The fabric agent selects the action $a_t$ based on the state $s_t$ given by the fabric computing environment and returns it to the fabric computing environment. The fabric computing environment selects the decision $a_t$ for user requirements based on $a_t$ and calculates the reward $r_t$ for the system. Based on the data sampling rate and density decisions of the service model to update its computational intensity, data size, transmission rate,and accuracy, the next state $s_{t+1}$ is obtained and this information is returned to the fabric agent. The IASC algorithm saves the tupe $(s_t, a_t, r_t, s_{t+1})$ to the experience pool, and during the training process, it determines whether the amount of data is sufficient. If the amount of data is sufficient, a batch of data is randomly sampled from the experience pool to train the actor network and critic network, and the parameters of the target network are softly updated. The IASC algorithm will get the optimal deployment decision $a_t^*$ that maximizes the measure $U_i^t$ by continuous training.

**Algorithm 1** Intelligent Adaptive Sensing and Computing (IASC) Scheme

**Require:** fabric $N$, DNN model $M$, number of iterations Episode, max steps of training $T_{\max}$, sampling weight coefficient $\beta$, discount factor $\gamma$

Initialize critic network and actor network parameter;
Initialize target network parameters using the online network;
Initialize prioritized experience replay buffer $R$ and set $p_1 = 1$
**for** episode: = 1,2,$\cdots$,Episode **do**
  Initialize environment to obtain the initial state $s$
  **for** $t = 1, 2, \cdots, T_{\max}$ **do**
    Fabric agent select an action based on state $s$:
    Update$A_j^t$,$\eta_j^t$, $r_j^t$, $c_j^t$, and obtain the next state $s'$
    Obtain the reward $r = \sum_{i=1}^{N} U_i^t = \sum_{i=1}^{N}\left(-w_i^L\ l_i^t + w_i^A A_i^t\right)$
    Store a new tuple $(s, a, r, s')$ in the experience reply buffer $R$
    **if** R is enough **then**
      Sample $(s, a, r, s')$ through prioritized experience replay
      Compute $y_i = r_i + \gamma Q^*\left(s', \mu^*\left(s'|\theta^{\mu^*}\right)|\theta^{Q^*}\right)$
      Compute importance sampling weight by Eq.(31) and Eq.(32)
      Update the critic by minimizing the loss: $L = \frac{1}{N}\sum_i w_i \left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)^2$
      Update the actor using the sampled policy gradient by Eq.(27)
      Soft update according to Eq.(29)
      Update priority by Eq.(30)
    **end if**
  **end for**
**end for**

## V. Performance Evaluation

In this section, we evaluate the performance of the IASC scheme. First, we introduce the set up of the experiment. Then, we give several comparison algorithms as the baseline. Finally, we analyze and discuss the performance of the algorithm and evaluate the impact of different parameters on the performance of the algorithm.

### A. Experimental Setup

In the experiment, we set the number of fabric sensors to 10, the number of DNN models for fabric data to 20, the size of the experience pool to 8,000, the number of sampling batches to 128, the soft update parameter to 0.01, and the reward discount factor to 0.99. For the algorithm, the actor networks have five layers, and the number of neurons in each layer is the size of the state space, 256, 512, 256, and the size of the action space, respectively. Similarly, the critic networks have five layers, and the number of neurons in each layer is the state space plus the size of the action space, 256, 512, 256, and 1, respectively. We set the learning rates of the actor and critic networks to $0.0001$ and $0.001$, respectively. We perform a total of $500$ iterations. In each iteration, when the number of training rounds reaches 300, the iterative process ends. The parameters of the network are given in Table I.

### B. Comparison Algorithm

To evaluate the effectiveness of the IASC algorithm, we choose to compare it with the following four baseline algorithms: random sensing and computing scheme, DDPG and upper confidence bound (UCB) [26] jointly optimized sensing and computing scheme, soft actor–critic (SAC)-based sensing and computing scheme, and twin delayed deep deterministic policy gradient (TD3)-based sensing and computing scheme. Then, we introduce these algorithms.

- Random sensing and computing scheme (i.e., Random): The computing model, sampling rate, and sampling density are randomly selected and placed on fabric agents; in other words, we randomly generate deployment decisions $a_t^*$ and compute system rewards $r_t$.
- DDPG and UCB jointly optimized sensing and computing scheme (i.e., DDPG-UCB): This scheme uses DDPG for

TABLE I
PARAMETER SETTINGS

| Parameters | Values |
|---|---|
| Number of fabric sensors | 10 |
| Number of DNN service models | 20 |
| Experience pool size | 8000 |
| Number of sampling batches | 128 |
| Soft update parameters | 0.01 |
| Reward discount factor | 0.99 |
| Learning rate for actor network | 0.0001 |
| Learning rate for critic network | 0.001 |
| Iteration number | 500 |
| Training rounds | 300 |

sensing and UCB for computing. Specifically, first, after calculating historical observations for each model through the UCB algorithm, we select $q_j$ models to form a state space $s_t$ and send it to the DDPG algorithm. The DDPG algorithm uses $s_t$ as the state input of the fabric agent and output $a_t$. Then, the fabric agent calculates the system's reward $r_t$ and the establishment of the placement at the time slot, and sends them to the DDPG algorithm and the UCB algorithm, respectively. The UCB algorithm enters the next iteration state and continues to select the next state $s_{t+1}$, which is sent to the DDPG algorithm.

- SAC sensing and computing scheme (i.e., SAC): The SAC algorithm is an off-policy algorithm based on maximum entropy reinforcement learning. The SAC algorithm maximizes the entropy while maximizing the expected reward. The larger the entropy, the more random the action output by the algorithm, so the SAC algorithm has a strong exploration ability.
- TD3 sensing and computing scheme (i.e., TD3): The TD3 algorithm doubles the number of each critic neural network and takes the one with the smallest output value of a pair of critic networks as the update target, which increases the stability of the algorithm. Moreover, TD3 adds a delayed update mechanism that can improve the critic update frequency. After the evaluation performance is stable, the critic network can make a more correct evaluation of the action to guide the actor to learn. TD3 adds noise to the actions input by the actor target network so that the actions are randomly distributed within a small range, making the estimation more accurate and preventing overfitting of the peak value in the value estimation.

### C. Performance Evaluation of IASC Scheme

*1) IASC Scheme Convergence Analysis:* We first analyze the convergence of the IASC algorithm, as shown in the Fig. 4. In this experiment, we perform 10 experiments and take the average of the experiments as the reward. In the Fig. 4, the shadow around the solid line is the area covered by the maximum and minimum values in the 10 experiments, which shows the fluctuation of the results. From the figure, we can see that the IASC scheme reward value fluctuation range keeps rising from 0 to 300 epochs. The reward value

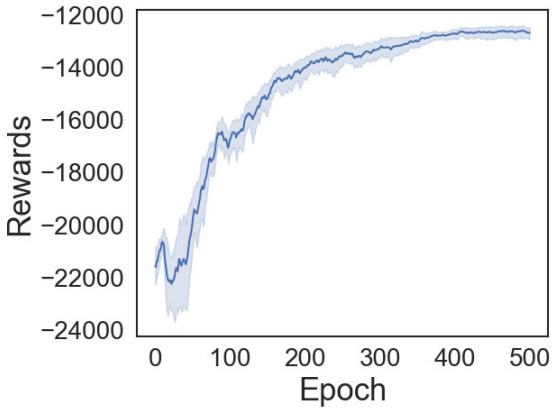


Fig. 4. The average reward of the IASC algorithm.

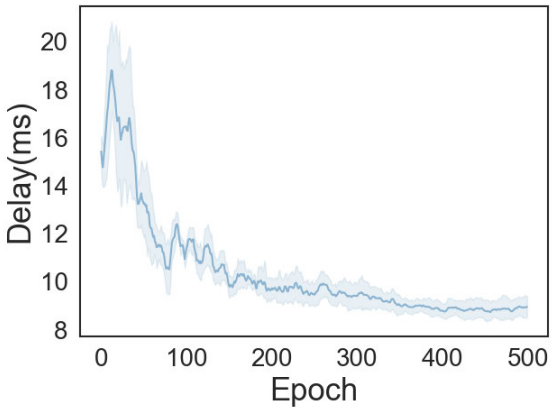


Fig. 5. The latency of the IASC algorithm.

fluctuates greatly before 200 epochs, and after the 200th epoch, the fluctuation is small, and it tends to be flat from the 300th epoch. The result shows that the IASC scheme gradually converges with the increase in the number of iterations.

Moreover, we give the latency of the IASC algorithm, as shown in Fig. 5. The curve distribution of the latency obtained by 10 simulation experiments is shown in the light blue area in the figure. The processing latency fluctuates greatly before the 200th epoch, and the fluctuation range gradually decreases with the increase in training epoch. After the 200th epoch, the processing latency gradually becomes smooth, and the fluctuation is small. The average latency is obtained by taking the average of the total delay of processing in these 10 experiments, which shows a downward trend as a whole and gradually becomes smoother. This also shows that the IASC algorithm can effectively reduce the latency.

Furthermore, Fig. 6 shows the curve distribution of the average accuracy. From the figure, we can see that before the 200th epoch, the average accuracy of the system fluctuates greatly, and then it is in a flatter state. The solid line is the average result of these 10 experiments, and the average final accuracy rate is 0.85. Thus, we provide the convergence analysis of the algorithm.

*2) Experimental Analysis of the IASC Scheme:* To verify that the IASC algorithm can adaptively adjust the sampling

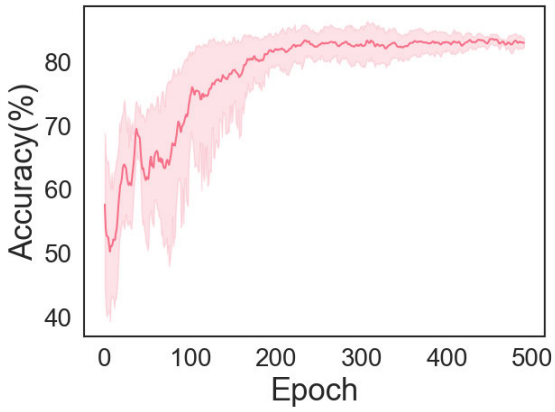


Fig. 6. The accuracy of the IASC algorithm.

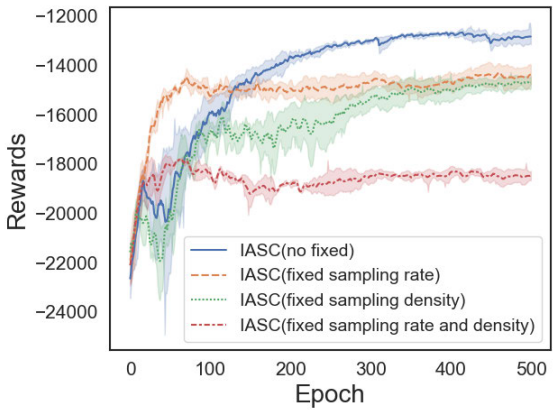


Fig. 7. The adaptivity of the IASC algorithm.

rate and density of fabric sensing data, in the experiment, we adjust the action space. Specifically, we adopt three methods: IASC with a fixed sampling rate and adaptive sampling density, IASC with a fixed sampling density and adaptive sampling rate, and IASC with a fixed sampling rate and density. Specifically, when the algorithm adopts the fixed sampling density and rate, the fabric agent assigns a fixed sampling rate and sampling density to the fabric sensor, which are recorded as $r^*_{sa}$ and $d_*$, respectively. Thus, the actions of the three groups of experiments are defined as follows:

- IASC (fixed sampling rate and adaptive sampling density): that is, $a_t = \{d_1^t, g_1^t, \cdots, d_M^t, g_M^t\}, \quad r^*_{sa}$
- IASC (fixed sampling density and adaptive sampling rate): that is, $a_t = \{r_{sa,t}^1, g_1^t, \cdots, r_{sa,t}^M, g_M^t\}, \quad d_*$
- IASC (fixed sampling rate and sampling density): that is, $a_t = \{g_1^t, \ldots, g_M^t\}, \quad r^*_{sa}, \quad d_*$

Fig. 7 gives the experimental results of the adaptivity of the IASC algorithm. From the figure, we can see that the IASC with a fixed sampling rate and density has the lowest reward. This is because in this case, the amount of data transmitted by the fabric is difficult to adapt to the network and model, resulting in high latency and low accuracy. From the figure, we can also see that in the case of IASC with a fixed sampling rate and adaptive sampling density, and IASC with a fixed sampling density and adaptive sampling rate, the final stable

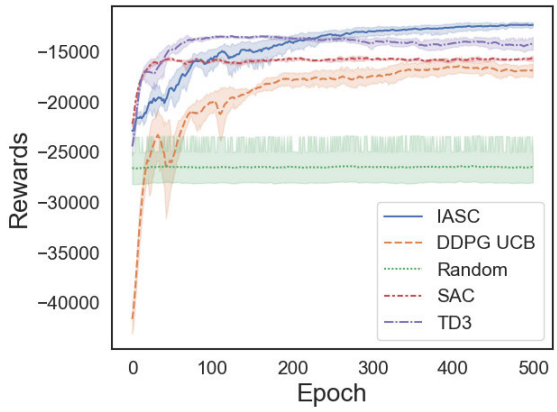


Fig. 8. Average reward under Random, SAC, TD3 DDPG-UCB and IASC algorithm.

level of the reward value curve is relatively close and lower than the IASC scheme. This shows that the IASC algorithm can improve the reward by adjusting the sampling rate and sampling density to find a suitable sampling rate and density.

*3) Comparative Results:* In this experiment, we compare the IASC algorithm with random sensing and computing scheme, DDPG and UCB jointly optimized sensing and computing algorithm, SAC sensing and computing scheme, and TD3 sensing and computing scheme, as shown in Fig 8. From the Figure, we can see that the IASC scheme has the highest average reward when it is finally stable, which also shows that our proposed method performs the best. The stable average reward of the TD3 scheme is lower than that of the IASC algorithm. The strategy of taking a lower Q value for the dual critic network makes the algorithm relatively conservative to a certain extent.

The stable average reward curve of SAC algorithm is lower than that of the TD3 algorithm. However, the SAC algorithm can quickly reach a plateau in the early stage, which is faster than the IASC algorithm and the TD3 algorithm. This is because of the strong search ability of the SAC algorithm, which can find strategies with higher reward values in the early epoch and continuously choose among these strategies in subsequent training epoch. In addition, the SAC algorithm discards strategies with low reward values, which could have higher rewards in the future for the current scenario. Therefore, the stable average reward of SAC algorithm in a flat state is lower than that of the IASC algorithm.

The final stable average reward curve of the DDPG-UCB scheme is lower than that of the SAC algorithm. The DDPG algorithm only focuses on the sampling rate and sampling density, which cannot have a comprehensive understanding of all model information. While the UCB algorithm pays attention to the model placement, changes in the sampling rate and density lead to changes in the computational intensity and accuracy of the model, so it may take more time for the UCB algorithm to evaluate the performance of the model based on historical observations.

*4) The Impact of the Number of Fabrics on IASC Algorithm:* To verify the algorithm performance in complex environments, in this experiments, we examine the number of fabrics on the

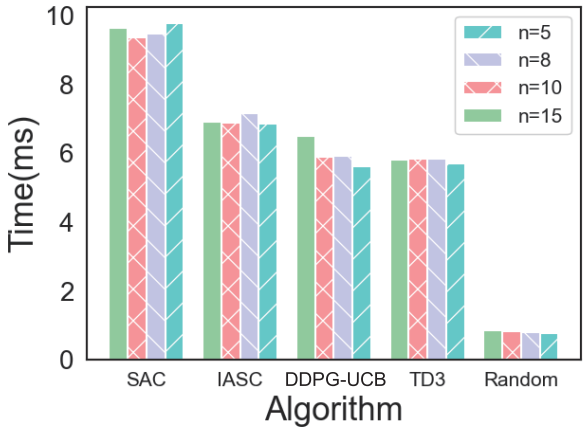


Fig. 9. Algorithm running time under different number of fabrics.

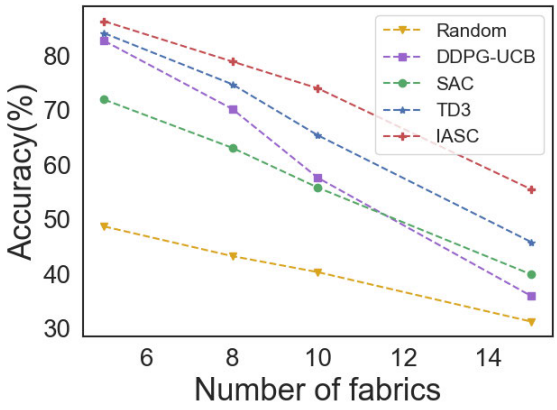


Fig. 11. System accuracy under different the number of fabrics.

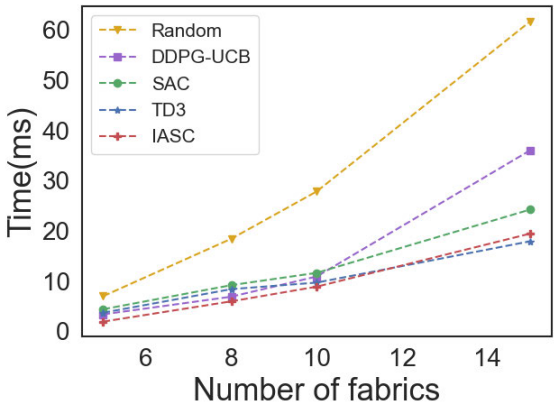


Fig. 10. System processing latency under different number of fabrics.

algorithms. We set the number of models as $m = 20$, set the number of fabrics as $n = 5, 8, 10, 15$, and divided them into four groups of experiments for training. Each group of experiments includes all of the comparison methods, and we take the total latency and system average accuracy at the end of the epoch as the result.

Fig. 9 shows the running times of different algorithms under different situations. From the figure, we can see that the SAC algorithm has the longest running time, followed by the IASC algorithm. The running time of the DDPG-UCB scheme ranks third because the decision time of the UCB algorithm is faster, and its state dimension and action dimension are smaller than those of the IASC scheme, which makes its running time better than that of the IASC algorithm. Moreover, the fewer number of fabrics, the faster the DDPG-UCB algorithm. The running time of the TD3 algorithm ranks fourth. The random strategy algorithm only randomly selects the strategy, so the running time is the fastest.

The system processing latency varies with the number of fabrics, as shown in the Fig. 10. From the figure, we can see that the total processing latency of the five algorithms increases with the increase in the number of fabrics. This is because consider the limited computing and communication resources of edge cloud, with the increase the number of fabrics, the fabric agent reduces the computing resources allocated to each model, which increases the communication transmission delay and computing delay. Moreover, the total processing delay of the random strategy algorithm is the largest in the five groups. When $n = 10$, the curve of the total processing delay of the four algorithms (i.e., TD3 algorithm, SAC algorithm, DDPG-UCB algorithm and IASC algorithm) is relatively close. Moreover, the delay of SAC algorithm is the highest, while the delay of IASC algorithm is the smallest. After the number of fabrics $n = 12$, the system processing delay of the TD3 algorithm is lower than that of the IASC algorithm, indicating that after the number of fabrics increases, the performance of the TD3 algorithm is better than that of the IASC algorithm, while the total processing delay of the SAC algorithm and the DDPG-UCB algorithm becomes higher. In summary, the IASC algorithm has the best overall performance in the system processing delay among the five algorithms.

Fig. 11 shows the curve of the average accuracy of the algorithm concerning the number of fabrics. It can be seen that the average accuracy of the system decreases with the increase the number of fabrics. This is because the algorithm reduces the calculation amount of the DNN model by reducing the sampling rate and density to balance the processing delay and the accuracy. This leads to the accuracy of the model processing decreasing. The random scheme algorithm has the lowest average accuracy curve. The TD3 algorithm is below the IASC algorithm and above the SAC algorithm, and when n=15, the average accuracy of the system is reduced to about 50%. This shows that the adjustment effect of the TD3 algorithm on the average accuracy rate is poor. The SAC algorithm is lower than that of the DDPG-UCB algorithm before $n = 11$, and it is relatively higher after n=11. However, when $n = 15$, the average accuracy of SAC and DDPG is reduced to less than 50%, so neither of these two algorithms is suitable for scenarios with a large number of users.

## VI. Conclusion

In this paper, we first propose a 6G fabric computing network which includes a fabric sensing layer, 6G communication layer, and fabric agent layer. Then, taking fabric sampling rate, sampling density, and model placement as variables, we have formulated an optimization model with the goal of

minimizing the user acquisition delay while ensuring accuracy. To solve this model, we have designed the IASC scheme based on the deep reinforcement learning algorithm. This scheme can achieve the optimal control of the data by adjusting the sampling rate and density of the fabric. By combining the placement of the intelligent model, the user's benefits can be maximized. Finally, we have carried out experiments to verify our algorithm's effectiveness. The experimental results show that compared with several baseline algorithms, our proposed algorithm can minimize the delay and maximize the accuracy. In future work, we will consider the energy consumption in 6G fabric computing.

## REFERENCES


[1] Y. Wu, T. Q. Duong, and A. L. Swindlehurst, "Safeguarding 5G-and-beyond networks with physical layer security," *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 4–5, Oct. 2019.

[2] V.-D. Nguyen, T. Q. Duong, and Q.-T. Vien, "Editorial: Emerging techniques and applications for 5G networks and beyond," *Mobile Netw. Appl.*, vol. 25, no. 5, pp. 1984–1986, Oct. 2020.

[3] S. Zang, W. Bao, P. L. Yeoh, B. Vucetic, and Y. Li, "Filling two needs with one deed: Combo pricing plans for computing-intensive multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 7, pp. 1518–1533, Jul. 2019.

[4] T. Q. Duong, K. J. Kim, Z. Kaleem, M.-P. Bui, and N.-S. Vo, "UAV caching in 6G networks: A survey on models, techniques, and applications," *Phys. Commun.*, vol. 51, Apr. 2022, Art. no. 101532.

[5] Y. J. Jeong, Y. E. Kim, K. J. Kim, E. J. Woo, and T. I. Oh, "Multilayered fabric pressure sensor for real-time piezo-impedance imaging of pressure distribution," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 2, pp. 565–572, Feb. 2020.

[6] T. Wang and S. Lin, "Negative impedance capacitive electrode for ECG sensing through fabric layer," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–8, 2021.

[7] M. Chen et al., "Fabric computing: Concepts, opportunities and challenges," *Innovation*, vol. 3, Oct. 2022, Art. no. 100340.

[8] A. F. Abouraddy et al., "Towards multimaterial multifunctional fibres that see, hear, sense and communicate," *Nature Mater.*, vol. 6, no. 5, pp. 336–347, May 2007.

[9] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 257–266.

[10] T. Liu, S. Ni, X. Li, Y. Zhu, L. Kong, and Y. Yang, "Deep reinforcement learning based approach for online service placement and computation resource allocation in edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 3870–3881, Jul. 2023.

[11] P. Lin, Z. Shi, Z. Xiao, C. Chen, and K. Li, "Latency-driven model placement for efficient edge intelligence service," *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 591–601, Mar. 2022.

[12] J. Jiang et al., "Joint model and data adaptation for cloud inference serving," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2021, pp. 279–289.

[13] Z. Xu, L. Zhou, S. Chi-Kin Chau, W. Liang, Q. Xia, and P. Zhou, "Collaborate or separate? Distributed service caching in mobile edge clouds," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 2066–2075.

[14] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1617–1655, 3rd Quart., 2016.

[15] D. Wang, D. Chen, B. Song, N. Guizani, X. Yu, and X. Du, "From IoT to 5G I-IoT: The next generation IoT-based intelligent algorithms and 5G technologies," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 114–120, Oct. 2018.

[16] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, Aug. 2019.

[17] J. Pan et al., "EdgeViTs: Competing light-weight CNNs on mobile devices with vision transformers," 2022, *arXiv:2205.03436*.

[18] X. Shi et al., "Large-area display textiles integrated with functional systems," *Nature*, vol. 591, no. 7849, pp. 240–245, Mar. 2021.

[19] S. Zeng et al., "Hierarchical-morphology metafabric for scalable passive daytime radiative cooling," *Science*, vol. 373, no. 6555, pp. 692–696, Aug. 2021.

[20] M. Chen et al., "Multifunctional fiber-enabled intelligent health agents," *Adv. Mater.*, vol. 34, no. 52, Dec. 2022, Art. no. 2200985.

[21] M. Chen et al., "Digital medical education empowered by intelligent fabric space," *Nat. Sci. Open*, vol. 1, no. 1, Jan. 2022, Art. no. 20220011.

[22] G.-Z. Yang and G. Yang, *Body Sensor Networks*, vol. 1. Cham, Switzerland: Springer, 2006.

[23] L. Yuan et al., "Tokens-to-token ViT: Training vision transformers from scratch on ImageNet," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 538–547.

[24] Y. Polyanskiy, H. V. Poor, and S. Verdu, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

[25] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[26] Y. Miao, Y. Hao, M. Chen, H. Gharavi, and K. Hwang, "Intelligent task caching in edge cloud via bandit learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 625–637, Jan. 2021.


**Yixue Hao** (Member, IEEE) received the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2017. He is an Associate Professor with the School of Computer Science and Technology, HUST. His current research interests include 5G networks, the Internet of Things, edge computing, edge caching, and cognitive computing.

**Long Hu** is an Associate Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), China. He was a Visiting Student with the Department of Electrical and Computer Engineering, The University of British Columbia, from August 2015 to April 2017. His research interests include the Internet of Things, software defined networking, caching, 5G, body area networks, body sensor networks, and mobile cloud computing.

**Min Chen** (Fellow, IEEE) is a Full Professor with the School of Computer Science and Engineering, South China University of Technology. He is also the Director of the Embedded and Pervasive Computing (EPIC) Laboratory, Huazhong University of Science and Technology (HUST). He was an Assistant Professor with the School of Computer Science and Engineering, Seoul National University, before he joined HUST. His Google Scholar Citations reached more than 39,000 with an H-index of 93. His top paper was cited more than 4090 times. He was selected as a highly cited researcher, from 2018 to 2022. He is a fellow of IET. He received the IEEE Communications Society Fred W. Ellersick Prize in 2017, the IEEE Jack Neubauer Memorial Award in 2019, and the IEEE ComSoc APB Outstanding Paper Award in 2022. He is the Founding Chair of IEEE Computer Society Special Technical Communities on Big Data. He is also the Chair of IEEE Globecom 2022 e-Health Symposium.