IEEE ComSoc | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS | IEEE Transactions on Machine Learning in Communications and Networking

Received 6 November 2022; revised 19 March 2023 and 24 May 2023; accepted 29 May 2023.
Date of publication 9 June 2023; date of current version 8 August 2023.

The associate editor coordinating the review of this article and approving it for publication was G. Yu.

Digital Object Identifier 10.1109/TMLCN.2023.3285171

Effective 3C Resource Utilization and Fair Allocation Strategy for Multi-Task Federated Learning

CHAOFENG ZHANG[1] (Member, IEEE), MIANXIONG DONG[2] (Member, IEEE), AND KAORU OTA[2] (Member, IEEE)

[1]School of Information and Electronic Engineering, Advanced Institute of Industrial Technology, Shinagawa, Tokyo 140-0011, Japan
[2]Department of Sciences and Informatics, Muroran Institute of Technology, Muroran 050-0071, Japan

CORRESPONDING AUTHOR: M. DONG (mxdong@mmm.muroran-it.ac.jp)

This work was supported in part by the JSPS KAKENHI under Grant JP22K17884, Grant JP20H04174, and Grant JP22K11989; in part by the Leading Initiative for Excellent Young Researchers (LEADER), MEXT, Japan; and in part by the JST, PRESTO, Japan, under Grant JPMJPR21P3.

ABSTRACT Nowadays, one of the main challenges in expanding AI applications is the effective use of Computation, Communication, and Caching (3C) resources. The complexity of the cloud environment and the diversity of resource usage make it challenging to complete federated learning tasks in a cost-effective, timely, and seamless manner. To address these issues, this paper proposes a comprehensive approach to optimize the overall service efficiency of federated learning in time-varying and 3C-constrained environments. Firstly, a utility function based on convergence efficiency is proposed to reflect the physical benefits of processing AI tasks. Then, a fair allocation strategy consistent with the optimization goal is designed by modeling the task allocation process through virtual queue Lyapunov drift. Next, a Federated Learning Long Short-Term Memory (LSTM) based Queuing Optimization and Allocation Policy Calculation Algorithm (FL-QAPC) is proposed for resource allocation policy calculation using multi-dimensional network state inputs with time series. This algorithm implements predictive control based on historical records. Finally, a feasible experimental test platform is conducted, which is extended to an actual wireless mobile scenario based on 5G. The superiority of the proposed solution is verified through comparison with other benchmarks.

INDEX TERMS Resource management, multitasking, learning systems, user centered design, mobile communication.

I. INTRODUCTION

WITH the explosive development of artificial intelligence (AI) applications, decentralized deep learning tasks are facing many new challenges. Federated learning (FL) has emerged as an adaptive and intelligent solution [1] to ensure the global effectiveness of the process through distributed computing, allowing the neural network to be updated through the collaboration of intelligent or edge devices [2]. This approach eliminates the need to upload security-sensitive data to the cloud. Instead, devices participating in the computation upload the trained results to the cloud and implement federated learning through data fusion [3]. This framework disperses the computing pressure of individual devices and protects users' privacy. Specifically, intelligent services using federated learning aim to address issues such as privacy leakage and insufficient computing power in distributed computing systems [4], achieving specific AI services, such as image recognition and behavior prediction, through distributed computing and parameter collaboration. The original data containing sensitive user information is not uploaded to the cloud; instead, local devices upload the locally trained weights or gradients. However, this also poses new challenges for communication, computation, and caching (3C) resources in the entire network system.

The enhancement of system capacity is typically achieved by increasing the consumption of physical 3C resources. However, in cloud-supported federated learning, the allocation policy may not be able to ensure that all local data is properly updated due to physical constraints on service

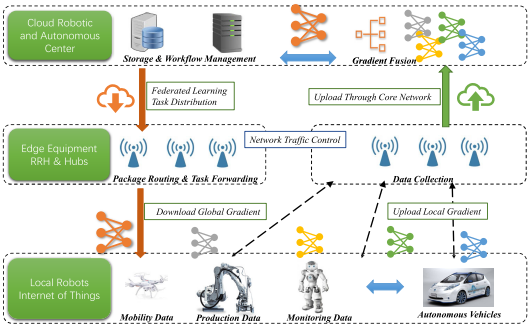


FIGURE 1. The data forwarding framework and scheduling of federated learning involves the distribution of tasks to each device through local edge equipment by the cloud. The local devices collect the original local training sets and update the assigned neural network model. The new gradients are then uploaded to the cloud for gradient fusion.

capacity. To address the issues, several studies have proposed optimization techniques. For instance, Khan et al. [5] consider the self-interest of local devices and introduce an incentive mechanism based on the Stackelberg game theory model to encourage their participation, which increases individual service capacity and overall utilization efficiency of the remaining capacity. In general, cloud systems require communication coordination among edges to optimize the learning process on each device. The edge devices act as buffers for devices in different workplaces and enhance multi-task processing efficiency.

The recent researches explore the integration of 3C resource optimization with specific federated learning applications. For instance, Wang et al. [6] attempted to use federated learning tasks to adjust popular information in memory by utilizing delay and energy consumption, included in the utility function, as reference indicators. Although they did not consider further optimizing the actual training effect of each round of federated learning, they mention that their reinforcement learning model was capable of effectively handling the distribution of established benefits. In a similar vein, Boura et al. [7] propose the efficient optimization of 3C resources by introducing local data fusion, particularly for updating local neural networks, which can be achieved by edge computing to reduce communication and energy consumption. The solutions proposed by early adopters demonstrate outstanding performance in handling single or one-time AI tasks. However, addressing multi-task processing with precise individual performance using federated learning requires consideration of more unique issues. Firstly, synchronous federated learning [8] involves multiple local devices performing gradient updates within a specified time, while the remaining idle resources can be used for other AI tasks generated during the same period. A potential challenge is how to make reasonable use of 3C resources to minimize the waste of computing and communication resources. Secondly, federated learning requires performing multiple global updates in sequence. The benefits of global updates are diminishing [9]. How to fairly and simultaneously process multiple AI tasks under resource constraints has not been well addressed. Thirdly, the optimization algorithm for multi-task federated learning [10] outputs an allocation strategy based on the current network state. However, demand surges or untransmitted information may cause the optimization plan to fail. A predictive control scheme based on historical states is needed to achieve a more stable control policy.

To address the unique challenges of multi-task federated learning in time-varying and 3C-constrained environments, this paper investigates multi-task assignment strategies to optimize the overall service efficiency of federated learning. First, by observing the actual physical meaning of the convergence efficiency of a single global update, we design a utility-based task processing benefit and set the overall optimization goal accordingly. Then, for the task allocation process, we model it through virtual queue Lyapunov drift and design a more fair allocation strategy consistent with the optimization goal. To achieve more reliable predictive control, we propose a FL Service oriented LSTM based Queuing Optimization and Allocation Policy Calculation Algorithm, which calculates optimized allocation strategies based on time-series network states while maintaining stability and maximizing the utility benefits of multiple tasks. Finally, we conduct simulations in a time-varying scenario to compare the convergence efficiency of multiple federated tasks using the proposed algorithm. The results show that the proposed solution not only maximizes task utility gains but also optimizes overall resource utilization.

The main contributions of this paper are highlighted as follows:

- Develop a utility function based on convergence efficiency to reflect the physical benefits of AI task processing and optimize the system accordingly.
- Design a fair allocation strategy consistent with the optimization goal by analyzing the Lyapunov drift of the AI task queue.
- Propose an algorithm for calculating resource allocation policies using multi-dimensional network state inputs with time series. This algorithm implements predictive control based on historical records.
- Conduct experiments on a feasible test platform and extend it to an actual wireless mobile scenario based on 5G. Verify the superiority of the proposed solution by comparing it with other benchmarks.

II. RELATED WORK

Federated learning updates the network model synchronously using the local wireless network, and multiple channels can be used simultaneously. However, the unstable network environment can pose significant challenges. Current research aims to develop novel solutions to optimize various aspects of federated learning's processing performance.

A. MULTIPLE TASK PROCESSING SYSTEM AND UTILITY QUANTIFICATION

As federated learning (FL) is increasingly used in various applications, Khan et al. [5] propose a task incentive mechanism based on the Stackelberg game to allow users to strategically set local iteration numbers to maximize their accuracy-related utility. Meanwhile, the base station (BS) acts as a leader and maximizes the utility of multiple tasks by setting global iteration numbers and accuracy levels in the global FL settings. Motivated by the goal of ensuring fairness and robustness, Hu et al. [11] design a multi-objective optimization federated learning algorithm, called FedMGDA+. A new algorithm is proposed to ensure convergence to the Pareto stationary solution. This method requires relatively few hyperparameters to be adjusted and does not sacrifice the performance of any participating users, making it well-suited for use as a processing system for multiple federated learning tasks. Zhang et al. [12] measure privacy loss and utility loss from the perspective of unified information theory. They emphasize that due to the inevitable trade-off between privacy and utility, federated learning algorithms need to optimize certain degraded utility to achieve sufficient privacy protection.

B. SYNCHRONOUS FEDERATED LEARNING OPTIMIZATION

To enable synchronous federated learning over wireless networks, Vu et al. [13] adopt multiple-input multiple-output (MIMO) and ensure stable operation during synchronous communication periods for each federated learning task. They minimize energy consumption by considering user requirements, time allocation, transmit power, and computing frequency, and use Fritz John and Karush-Kuhn-Tucker solutions to achieve stable convergence. This method also demonstrates that synchronous federated learning can achieve good communication and stable convergence. Wang et al. [14] propose the Favor algorithm, an experience-driven control algorithm that intelligently selects client devices to participate in each round of federated learning, to counteract bias introduced by non-IID data and accelerate convergence. They also propose a mechanism based on deep Q-learning, which learns to select a subset of devices in each communication round to maximize rewards that encourage improved validation accuracy and penalize the use of more communication rounds.

C. CLOUD-EDGE-DEVICE NETWORKING BASED FEDERATED LEARNING

Wang et al. [14] propose model partitioning and task scheduling to minimize time costs and avoid wasting computing resources, in order to fully utilize computational resources on terminal devices and edge servers. To address the NP-hard networking problems, they develop the Federated Learning Offloading Acceleration (FLOA) algorithm to obtain suboptimal solutions. They also design a task offloading method based on matching theory to achieve resource allocation for collaboration among cloud, edge, and local devices.

D. MULTIPLE RESOURCE OPTIMIZATION

The optimization of multiple resources in federated learning is a non-convex problem and is usually divided into several sub-problems for optimization. Dinh et al. [15] address the trade-off between model and energy consumption convergence by formulating it as a non-convex problem with three sub-problems. They investigate the impact of local training rounds on global parameter updates and convert it into a trade-off problem between energy consumption and convergence rate. By optimizing the three sub-problems of communication energy consumption and calculation energy consumption, the proposed method outperforms benchmarks in terms of convergence rate and accuracy.

III. SYSTEM FRAMEWORK AND FLOW CUTTING

In this section, we provide an overview of the local and global updating procedures for a cloud-supported federated learning (FL) system. At first, we describe the local and global updating procedures in the mathematical model. Additionally, we propose a convergence-based utility function to evaluate the learning performance and establish the optimization goal for the overall task processing framework.

A. SYSTEM STRUCTURE

As shown in Fig. 1, the framework comprises three main components: a cloud center for data fusion, edge devices for task or model data forwarding [16], and local devices for sample collection and local updating.

- **Cloud:** Cloud server is responsible for aggregating the data uploaded by local devices, typically the weights or gradients of the neural network model after local training.
- **Edge:** Devices located at the edge layer are tasked with monitoring and managing available network resources. By observing and predicting channel conditions, computing resources, and energy consumption, edge devices forward new AI tasks and collect trained data.
- **Local:** Local devices are used for local updating. The local updating process involves downloading global weights, performing local training, and uploading newly trained gradients.

1) CLOUD AND DEVICES

We consider a federated learning system consisting of a cloud center for data collection and fusion at the top layer and an edge layer comprised of M base stations (BS) that connect with the cloud center, acting as edge nodes for forwarding tasks and collecting data [17]. Correspondingly, $|N|$ local devices are managed by these edge nodes, denoted as $N = \{1, 2, \cdots, n, \cdots\}$. During the predefined time frame $T = \{1, 2, \cdots, t, \cdots\}$, there are $|\Phi|$ applications requiring federated learning computing services denoted as $\Phi =$

TABLE 1. List of Symbols

Symbol	Description
n, N	Local devices for local data training.
t, T	Time slot and time frame.
Φ	Task set for federated learning.
q, z	Difficult and intrinsic importance of tasks to achieve convergence threshold.
φ	Size of the training model.
w^0, w^*	Weights of the training model at start and final.
$\phi_{n,k}$, ϕ'_k	Number of the datapoints.
z_k, q_k	Original importance value and calculation difficult of task k.
c_n	Computation ability of device n.
$\mu_{1,n}$, $\mu_{2,n}$	Download and upload bandwidth of device n.
$t_{1,n}$, $t_{2,n}$, $t_{c,n}$, t_n	Time cost of each local processing step and the whole process.
P_{l_k}, P_{g_k}	Number of local and global rounds of task k.
$\bar{U}_{r,n,k}$, $U_{r,k}$	Utility gain of global round r for assigned task k.
ϵ	Convergence threshold to complete the tasks.
Γ	Learning efficiency of each global round.
ξ, γ	Learning rate and preset approximation rate of each global round.
Q_n, Qu_n	Virtual backlog and utility backlog of device n.
A_n, Au_n	Assigned backlog of device n.
W_n, Wu_n	Processing ability of device n.
R_n	Reward value if assigned to device n.

$\{1, 2, \cdots, k, \cdots\}$. The notations used in the following discussion are listed in Table 1.

2) TASK PROCESSING

Using the general setup, the raw data collected by local devices may be different, which is considered as independent and identically distributed (IID) data in the overall learning process. To ensure fairness and convergence rate, the system requires at least Φ_k devices for local training, corresponding to a specific training task $k \in \Phi$. The k-th training model has three common attributes: computation difficult q, model size φ, and importance z. Firstly, we define the computation difficult of task k as,

$$q_k = F(w^0) - F(w^*), \tag{1}$$

where w^0 represents the initial state of the training model weights, and w^* is the converged state of the weights. The value $F(w^*)$ is typically acquired from previous training experiences with similar tasks, and is used to define the training difficulty q_k as a predefined constant. The physical meaning of q_k is the distance between the initial loss value and the convergence loss value. The range of values is within 0 to 1, using binary cross-entropy. Generally, the difficulty of convergence is closely related to the initial model. In extreme cases, the loss value after convergence is 0. Therefore, in practical applications, $F(w^*)$ is sufficiently small and $F(w)$ is randomly initialized, the more appropriate initial weights and distribution of the training samples can effectively reduce the computational difficulty.

The function $F(w)$ is calculated by

$$F(w) = \frac{1}{\phi'_k} \sum_{n \in N} \phi_{n,k} f_n(w) \tag{2}$$

where f_n is the corresponding loss function produced by device n, and $\phi_{n,k}$ is the number of datapoints for local training. ϕ'_k denotes the total number of datapoints. The weights calculated by the local devices are partly contributed to the final fusion process according to the number of local datapoints.

Considering the reality, we assume the task importance, denoted by

$$Z = \{z_1, z_2, \cdots, z_k, \cdots\}, \tag{3}$$

which indicates the inherent priority of the applications. Task importance is a measure of the relative importance of each task in a multi-task learning problem. It can guide how a model allocates resources and weights to optimize overall performance. Some tasks may require more resources and effort to solve than others, and task importance can be used to select the optimal combination of tasks and allocate resources accordingly. For instance, tasks related to driving or rescue operations may have an importance value of 4, while botanical identification tasks may have an importance value of 1. If all tasks are assigned an importance value of 1, the allocation of resources for all tasks is considered to be fair. The procedure of how to estimate the task importance can follow some practical operation manual, e.g. technical rescue (K9 Search and Rescue: A Manual for Training the Natural Way).

3) DOWNLOAD AND UPLOAD

To describe the output capacity of device n, its computation ability is denoted as c_n. For any time $t \in T$, the device's download bandwidth is recorded as $\mu_{1,n}$, and its upload bandwidth is recorded as $\mu_{2,n}$.

Correspondingly, as shown in Fig. 2, when a task is scheduled to end device n, the time cost is calculated after completing the following three phases [18]: downloading phase $t_{1,n}$, local training phase $t_{c,n}$, and uploading phase $t_{2,n}$. The download time $t_{1,n}$ is denoted as:

$$t_{1,n} = \frac{\varphi_k}{u_{1,n}}, \tag{4}$$

where φ_k is the model size. Similarly, the upload time $t_{2,n}$ is calculated as

$$t_{2,n} = \frac{\varphi_k}{u_{2,n}}. \tag{5}$$

The time to complete a single loss function calculation and back-propagation is denoted as,

$$t_{c,n} = \frac{\phi_n \varphi_k}{c_n}. \tag{6}$$

Before uploading data for global data fusion, the local device is required to perform P_{l_k} local rounds. For any end device i, the total time required to complete one global training is given by:

$$t_n = P_{l_k} t_{c,n} + t_{1,n} + t_{2,n}. \tag{7}$$

For a given task, the completion utility $U_{r,k}$ in global round r is defined as the metrics that reflect the quality of the trained model, which discussed in the following subsection. The average utility gain $\bar{U}_{r,n,k}$ for an end device n completing r-th global round is defined as follows.

$$\bar{U}_{r,n,k} = \frac{\phi_{n,k} U_{r,k}}{\phi_k' t_n}. \tag{8}$$

The specific definitions of local update round P_{l_k} is discussed in the following subsection.

B. LOCAL COMPUTATION AND DATA FUSION

The primary objective of the allocation policy is to coordinate the order of tasks Φ and the responding devices i to achieve optimal task completion performance, including utility gain. However, quantifying the benefit of each federated learning (FL) round is a challenging task. In this subsection, we propose a utility function $U_{r,k}$ as a common evaluation standard to describe the received utility reward after completing the r-th global round update. Accordingly, we analyze the relationship between the convergence rate and the utility gain after each global updating round.

1) CONVERGENCE AND LEARNING EFFICIENCY

There is a trade-off between the number of global rounds and local rounds in federated learning. In order to achieve the final convergence threshold ϵ, each task needs to go through a total of $P_l \cdot P_g$ local iterations, where P_g is the number of global updating rounds. A global update occurs when the cloud completes one round of data collection from the local devices and merges the gradients uploaded by the local devices.

The global update function for the weights are supposed to be,

$$w^t = \frac{1}{\phi_k'} \sum_{n=1}^{N} \phi_{n,k} w_n^t, \tag{9}$$

where w_n is the updated weights. Instead, the uploading uses the fusion of gradients, which is formulated as,

$$\nabla F^t = \frac{1}{\phi_k'} \sum_{n=1}^{N} \phi_{n,k} \nabla F_n(w_n^t), \tag{10}$$

where ∇F_n is the gradient vector uploaded by device n. After P_g global updates, the final convergence condition holds,

$$F(w^t) - F(w^*) \leq \epsilon \tag{11}$$

where w^* is the final state of convergence. To explore the performance bound of global update, using the similar assumption of the previous paper [19], we assume the following two assumptions:

Assumption 1: The loss function $F(w)$ is β-strongly convex.

In neural networks, β-strong convexity refers to a property of the loss function $f(w)$ that satisfies the following inequality: $F(w) - F(w^*) \geq \frac{\beta}{2}|w - w^*|^2$, where w and w^* are two points in the parameter space and $|\cdot|$ denotes the Euclidean norm. This inequality implies that for any w, the difference between its function value and the optimal value is greater than or equal to a positive constant $\frac{\beta}{2}|w - w^*|^2$, where the larger the constant, the stronger the convexity.

In neural networks, strong convexity is a property often used to prove the convergence rate of optimization algorithms. When the loss function $F(w)$ is β-strongly convex, optimization algorithms such as gradient descent can converge faster towards the global optimal solution.

Assumption 2: The loss function $F(w)$ follows L-smooth.

In neural networks, L-smoothness refers to the property that the derivative (gradient) of the loss function is L-Lipschitz continuous, where L is a positive constant. Specifically, for any two points x and y, the difference between their gradients does not exceed L times their distance, i.e., $|\nabla F(x) - \nabla F(y)| \leq L|x - y|$. This is a very useful property as it guarantees that the step size and direction of each iteration in optimization algorithms such as gradient descent are stable.

Then, for the specific loss function $F_n(w)$ for device n, the condition number of the matrix is considered as L/β. Because of better stability, the curvature changes of the Hessian matrix $H_f(w)$ are usually smooth, which means that optimization algorithms set appropriate step sizes to achieve stable convergence boundaries. Therefore, we define the linear convergence bound as,

$$F(w^t) - F(w^*) \leq (1 - \Gamma)^t (F(w^0) - F(w^*)), \tag{12}$$

where the global convergence indicator Γ is

$$\Gamma = \frac{2\xi(2(\gamma - 1)^2 - (\gamma + 1)\gamma(3\xi + 2)\rho^2 - (\gamma + 1)\xi\rho^2)}{2\rho - 2(1 + \gamma)^2\xi^2\rho^3}. \tag{13}$$

The efficiency Γ is usually considered as [0, 1]. The parameter γ is the preset approximation rate, which satisfies the gradient descent of the surrogate function J_n^t, where we have

$$|\nabla J_n^t(w_n^t)| \leq \gamma |\nabla J_n^t(w_n^{t-1})|. \tag{14}$$

In addition, the hyper-learning rate ξ is the learning rate used for the cloud to perform global gradient updating. It is used to adjust the update step size for each parameter. The size of the hyper-learning rate usually affects the convergence speed and stability of the network. If the learning rate is too small, the training time will be longer, and if the learning rate is too large, the network may experience unstable oscillations during the training process.the parameter ξ is the global learning rate of federated learning. Generally, $\xi = 0$ means all the changes of gradients are uploaded to the server accurately. For example, ρ is the condition number calculated by the hessian matrix in [20].

2) CALCULATION ROUND AND UTILITY DEFINITION

Correspondingly, the number of global updates P_g is denoted as,

$$P_g = \frac{1}{\Gamma} log \frac{F(w^0) - F(w^*)}{\epsilon}. \tag{15}$$

The necessary global ground is mainly related to the efficiency of model learning Γ, the initial weights w^0, and the convergence threshold ϵ. P_l is the training round completed locally before each global update, denoted as

$$P_l = \frac{2}{\nu_1} log \frac{\nu_2}{\gamma}. \tag{16}$$

The parameter $\nu_1 \in (0, 1)$ is a weightage parameter used to balance the influence of global model updates and local model updates during the federated learning process. And the condition related parameter $\nu_2 \in (0, 1)$ is also constraint by the $J(z^\zeta) - J(z^*) \leq \frac{\beta}{L} \nu_2 (1 - \nu_1) \left(J(z^0) - J(z^*)\right)$, where the $J(z^0)$ and $J(z^\zeta)$ is the surrogate function at the local updating of the initiate state and the ζ-th iteration. $J(z^*)$ is the optimal solution for the local training. The surrogate function typically aims to preserve the main features and properties of the original function as much as possible, and produce similar computational results while reducing computational complexity.

C. SYSTEM OPTIMIZATION GOAL

In this subsection, we will briefly discuss the challenges of system optimization and how to set the optimization objectives of the system in FL tasks. Generally, in FL tasks, it is difficult to estimate the convergence rate after a round of global updates, and it is challenging to find appropriate optimization algorithms in multi-FL task environments due to the following three reasons:

- Due to the stochastic nature of the convergence process, it is difficult to precisely control the actual convergence effect by setting the number of local training iterations and learning rates to achieve the desired convergence effect after a single global update.
- There is a tradeoff between global and local updates. Excessive local learning may be counteracted by global updates, while frequent global updates may result in increased resource consumption and service latency.
- As the first challenge revealed, the benefits of training updates are not linear. Initial global updates may bring greater convergence benefits and accuracy improvements.

Based on the aforementioned challenges, we redesign the utility function to describe the benefits of a single global update: the utility obtained by task k after the r-th global round is,

$$U_{r,k} = [(1 - \Gamma)^{t-1} - (1 - \Gamma)^t] z_k q_k, \tag{17}$$

where r represents the r-th global round, z_k represents the inherent importance, and q_k is the calculation difficult. For instance, in a disaster monitoring scenario, the importance level is defined as the discrete safety level; in an autonomous driving system, the importance level is the importance of object recognition, including signal lights, pedestrians, and railings.

Here, we assume that only synchronous training based on IID data was used in this scenario. This is because asynchronous federated learning results in unequal weighting of gradients during fusion, leading to unpredictable convergence results each round. The same issue exists with Non-IID data, where the special nature of samples leads to significant differences in local training models, making the final loss function not universally representative. As a result, the controllability of convergence cannot be ensured through the rounds of local training and global updates. In contrast, the $f(w)$ of IID data typically satisfies the conditions of L-smoothness and β-strong convexity. In the case of IID data, the data distribution is the same for each device, so the gradients update for each device are also the same, and the gradients of the entire dataset can better satisfy the requirements of smoothness and convexity. Therefore, it is relatively easier to optimize when performing data fusion.

1) MEANING OF UTILITY

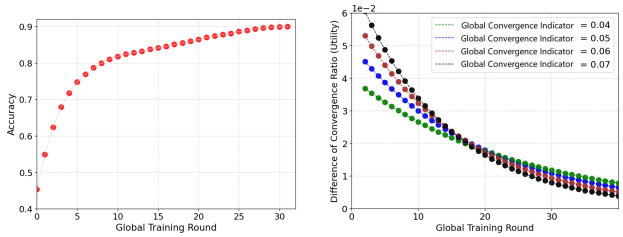
To address the challenge that the convergence effect is difficult to control precisely, the $[(1 - \Gamma)^{t-1} - (1 - \Gamma)^t]$ term is referring to the actual convergence rate of Γ in Eq. (12). The setting of Γ can follow Eq. (13), which can be obtained by approximating ratio γ and global learning rate ξ. When $\gamma \in (0, 1)$ and $\xi \in (0, 1)$ are small, then $\Gamma \in (0, 1)$. Therefore, it is possible to control the expected convergence of a single global update $(1 - \Gamma) \in (0, 1)$.

At the same time, we avoid the optimization complexity problem caused by the trade-off between global and local updates by setting a fixed Γ. For a given Γ, local devices adaptively adjust the number of updates and learning parameters to calculate the combination with the lowest energy consumption.

Obviously, the benefit of each global update are not linear. Eq. (17) describes the relationship between the training benefits and the convergence effect.

Fig. 2(a) shows the accuracy curve tested using the MNIST dataset. Here, $\Gamma = 0.05$, which means that the convergence efficiency of each round of global update is above 0.05. Correspondingly, the increase in accuracy is quite significant in the first 10 rounds. Then, the gain in accuracy per round is gradually decreasing.

Similarly, the y-axis in Fig. 2(b) represents the difference in convergence rates, which is calculated using Eq. (17) when $z_k = 1$. It can be observed that the rate of return on utility decreases as the number of calculation rounds increases. It is evident that the relationship between convergence rate and accuracy is not linear. Recent research has demonstrated that as the loss function decreases, the accuracy shows a certain upward trend, but different optimization algorithms correspond to different curve shapes [21]. Therefore,



(a) Prediction accuracy changes with global updating round.

(b) Difference of convergence ratio changes with global updating round.

FIGURE 2. The relations of accuracy, convergence rate and updating round.

we propose Eq. (17) to combine the mathematical meaning of convergence rate with the physical meaning of prediction accuracy.

2) SYSTEM OPTIMIZATION GOAL

During the time frame T, there are a total of $|N|$ devices located in $|M|$ edges that jointly serve $|\Phi|$ federated learning tasks to achieve weight convergence. However, in a time-varying wireless scenario, various unexpected conditions, such as device mobility, energy consumption problems, and unstable fading channels, make it challenging to calculate an optimal allocation policy within polynomial time. As a result, some of the tasks may not be trained enough that converge to the threshold ϵ.

Taking into account the constraints on communication, computation, and caching resources, the optimization objective of the system is to achieve optimal global utility gains within a limited time T, while simultaneously performing global updates for as many AI tasks as possible,

$$\max \qquad U = \sum_{t\in T}\sum_{k\in\Phi}\sum_{n\in N} \bar{U}_{r,n,k} \cdot X_{n,k,t}, \tag{18}$$

where $X_{n,k,t}$ is the optimization variable whether device n process the task k in time t. The optimization algorithm is responsible for optimizing the device switching at time slot t to achieve better utility gains. Here, $\bar{U}_{r,n,k}$ denotes the average utility gain of the r-th round of task k for device n.

The optimization goal mentioned above is an NP problem. Using brute force search for the algorithm leads to polynomial time complexity [22]. For each time slot t, the algorithm arranges the next task by complexity $O(k)$ ($k \leq |\Phi|$). Then, to determine the allocation policy among candidate local devices, the complexity is greater than $O(n^m)$. At most, (n/m) tasks are arranged for each t. Consequently, the overall complexity of the algorithm is at least $O(k^{(n/m)}n^m)$, making it unsolvable in polynomial time. Therefore, instead of using the unachievable brute force searching, we use the Lyapunov drift theory to quantify the allocation plan and train an LSTM-based deep learning allocation algorithm to determine the allocation vector.

IV. OPTIMIZATION ANALYSIS BASED ON STEADY-STATE MODEL

This section aims to analyze the impact of the one-step allocation plan, and we introduce the virtual queuing theory to describe the current state of the network. Then, we conduct mathematical deductions and derive a universal allocation policy based on the Lyapunov stability.

A. VIRTUAL QUEUE

Before discussing the real scenarios, we assume that the storage of each device is infinite. The current state of all the devices is described in an infinitely long virtual queue matrix Q, denoted as $Q(t) = \{Q_n(t), n \in N\}$. The dimension of Q is $|N|$. If a task k is allocated to the queue of a specific device n, the length of Q_n increases by the length of the task. For any time slot t, the length of tasks assigned to the queue is denoted as $A(t)$, and the length of tasks that are going to be processed is denoted as $W(t)$. Then, at time slot $t+1$, the new queue backlog of device n is calculated as follows,

$$Q_n(t+1) = Q_n(t) + A_n(t) - W_n(t) \quad \text{where} \quad \forall n \in N. \tag{19}$$

After completing any task, the device outputs the task utility $U_{r,k}$. We use the utility queue $Qu_n(t)$ instead of the corresponding task queue $Q_n(t)$ to describe the dynamic changes in the queue,

$$Qu_n(t+1) = Qu_n(t) + Au_n(t) - Wu_n(t) \tag{20}$$

where $Au_n(t)$ denotes the actual utility of tasks that are allocated to device queue n at time t, while $Wu_n(t)$ denotes the utility of tasks that are processed by the device at time t. It is important to note that if t is the minimum unit time (with the same unit as in Eq. (4), then Wu_n can be calculated by the average actual utility processing rate $\bar{U}_{r,n}$ in Eq. (8).

B. VIRTUAL UTILITY QUEUE

Using Lyapunov stability to ensure fairness in all queues is a common approach to improve queue efficiency. Typically, the number of tasks or the length of the task stack in the queue is used as a measure to optimize queue length and throughput [23]. However, to optimize the throughput of task value as described in Eq. (17), we adopt a utility queue instead of a traditional queue based on the physical length of the task. This prioritizes tasks with higher utility, meaning that urgent tasks requiring performance improvement are processed first, reflecting a better utility-cost ratio. Fig. 3 illustrates the different performance of local devices in processing tasks based on different queue metrics over a short period of time. The blue curve represents utility-based allocation, orange is processed in order, and green is averaged based on actual task processing time. The horizontal axis represents time, and the vertical axis represents the total utility obtained from processing tasks. It is evident that using utility as a metric for task allocation can effectively achieve better utility throughput. Therefore, we innovatively adopt the utility queue instead of

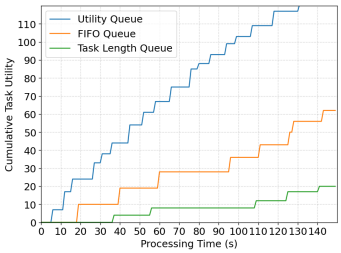


FIGURE 3. The graph illustrates the cumulative task utility gained after processing each task, where the blue line represents the utility-based allocation, which outperforms the classical queue length metric in terms of utility gain when processing tasks from the candidate pool.

the task stack queue to achieve a better and fairer optimization effect.

C. LYAPUNOV STABILITY

Similar to the virtual queue, the increase of length is influenced by the allocation policy $Au(t)$ in the utility domain. To make sure that the queue matrix $Qu(t)$ is stable, the overall allocation policy A_u should satisfy

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in N} Qu_n(t) < \infty. \tag{21}$$

Finally, to determine if the system has completed all the allocated federated learning tasks in Φ, we need to ensure that

$$\lim_{t\to\infty} \frac{Qu(t)}{t} = 0 \quad \text{where} \quad \forall n \in N, \tag{22}$$

where $Qu(t)$ denotes the backlog in the virtual queue at time t. If this condition is satisfied, we can say that the federated learning task processing system is Lyapunov stable.

To estimate the overall network state and explore an optimized solution, we can introduce the Lyapunov function $L(Qu(t))$ to analyze the backlog's influence in the virtual queue [24]:

$$L(\tilde{Q}u(t)) = \sum_{n\in N} \tilde{Q}u_n^2(t). \tag{23}$$

Correspondingly, in the next time slot $t+1$, the bound of this drift function becomes

$$\begin{aligned}\tilde{Q}u(t+1)^2 &\le (\tilde{Q}u(t) + Au(t) - Wu(t))^2 \\ &= \tilde{Q}u^2(t) + Au^2(t) + Wu^2(t) - 2\tilde{Q}u(t)Wu(t) \\ &\quad + 2\tilde{Q}u(t)Au(t) - 2Wu(t)Au(t),\end{aligned} \tag{24}$$

where $Au(t) \ge 0$ and $Wu(t) \ge 0$. In arbitrary step t, the conditional Lyapunov drift is

$$\begin{aligned}L(\tilde{Q}u(t+1)) - L(\tilde{Q}u(t)) &= Au^2(t) + Wu^2(t) \\ &\quad + 2\tilde{Q}u(t)Au(t) - 2\tilde{Q}u(t)Wu(t).\end{aligned} \tag{25}$$

D. BOUND OF LYAPUNOV DRIFT

Considering apply an arbitrary allocation policy $\pi \in \Pi$, we have the upper bound of the Lyapunov drift as

$$\begin{aligned}\Delta_\pi(t) &= (L(\tilde{Q}u(t+1)) - L(\tilde{Q}u(t))|\tilde{Q}u(t)) \\ &\le B + 2\sum_{n\in N} \tilde{Q}u_n(t)\mathbb{E}(Au_n^\pi(t)|\tilde{Q}u(t)) \\ &\quad - 2\sum_{n\in N} \tilde{Q}u_n(t)\mathbb{E}(Wu_n^\pi(t)|\tilde{Q}u(t)),\end{aligned} \tag{26}$$

where B is a constant, subject to

$$B = \sum_{n\in N}(\mathbb{E}(Au_n^\pi(t))^2 + (\mathbb{E}(Wu_n^\pi(t))^2) \le Au_{max}^2 + d, \tag{27}$$

and d is the bound of system processing ability.

Based on Eq. (26), and in order to compress the backlog, the system optimizes each product $\mathbb{E}(Qu_n(t)Au_n(t))$ of each queue separately. Therefore, one solution is to minimize the allocation cost, denoted as:

$$\arg\min \quad Cost = \sum_{n\in N} Qu_n(t)Au_n(t). \tag{28}$$

Based on the above deduction, in order to obtain a smaller $Cost$, the allocation policy should focus on assigning new incoming tasks to shorter queues, resulting in relatively lower allocation costs.

Then, the last part of Eq. (26) is to consume the backlogs and resulting in to lower bound, which the system optimizes the processing profit as,

$$\arg\max \quad Profit = \sum_{n\in N} Qu_n(t)Wu_n(t). \tag{29}$$

E. COORDINATION CHALLENGE FOR INDIVIDUAL ASSIGNMENT

Based on the above conclusion, it is recommended to assign longer queues to devices with higher computation capacity, which leads to higher profit outcomes. The utility processing ability $Wu(t)$ is roughly linearly related to the computation capacity. However, due to the philosophy of the $Cost$ function, new arriving tasks should be assigned to empty queues rather than to queues with powerful processing ability. To resolve the potential conflicts between Eq. (28) and Eq. (29), we define the revenue value as follows:

$$\begin{aligned}\arg\max \quad Reve &= \sum_{t\in T}\sum_{n\in N} Prof_n - Cost_n \\ &= \sum_{t\in T}\sum_{n\in N} Qu_n(t)Wu_n(t) - Qu_n(t)Au_n(t)\end{aligned} \tag{30}$$

which minimizes the upper bound of drift function in Eq. (26). Lower bound results in higher throughput performance, which directly influences the optimized value of the overall utility in Eq. (18).

F. POTENTIAL TRADEOFF WITH THE FUTURE

Once a global update is completed, the task enters the assignment pool again for the next round, creating a trade-off between current revenue outcomes and longer-term benefits when assigning candidate devices. The allocation policy must intelligently decide whether to prioritize achieving more revenue for individual devices based on the current queue state Qu [25], or to coordinate devices to quickly complete a round of global updates and renew the assignment pool. One classic approach is to use the Stackelberg model, where the system posts tasks in order of utility and assigns them to devices that gain more processing profit based on the current network state.

Calculating an optimal solution over the time frame T is also a challenge due to the computational complexity, which is not feasible for real-time online allocation using polynomial time. Therefore, we propose an online deep learning mechanism to optimize the allocation decision. However, the learning process compresses the computational difficulty of the NP hard problem by sacrificing accuracy. Thus, a deep learning-based allocation policy calculation frame with flexibility and forward-looking is required for this study.

V. LSTM BASED FEDERATED LEARNING ALLOCATION ALGORITHM

This section discusses the application of neural networks to output an optimized allocation policy. First, an LSTM-based model is used to process the multi-dimensional input of network states. Then, based on the conclusion in Section IV, the recommended value for each allocation policy is designed as the output of our proposed LSTM-based network framework. Finally, to make it practical and applicable online, an online allocation policy calculation algorithm is designed to handle potential changes in network capacity.

In the context of practical FL task control and distribution, control strategies based on current network states may be adversely affected by network congestion, which can result from various factors such as insufficient bandwidth, channel interference, and insufficient storage and forwarding devices. These factors may cause information loss or delay, which can directly lead to a lack of critical information necessary for control strategies to continue. To mitigate this issue, we propose using an LSTM-based predictive control solution.

A. LSTM FOR PREDICTIVE CONTROL

LSTM is a neural network that is well-suited for predictive control. In comparison to traditional RNNs, LSTM has better memory and information flow control capabilities, which enables it to handle long sequence data more effectively. Moreover, LSTM can mitigate issues such as gradient explosion and vanishing that arise from processing long-term sequence data. The predictive network based on LSTM, as illustrated in Fig. 4, can achieve predictive control without relying on the current network state. Even if information for the current time slot is unavailable due to network congestion

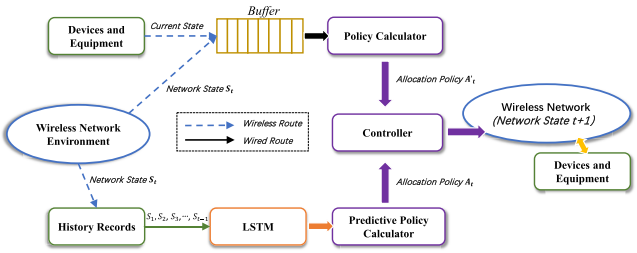


FIGURE 4. LSTM-based predictive policy controller. The system uses historical records to predict and calculate the allocation plan, which compensates the potential calculation delays or traffic congestion about current network states and local resources.

or other reasons, the system can still utilize historical data to predict and compute the allocation plan for the next time slot. This greatly enhances the reliability and stability of the system.

At the same time, the control scheme based on LSTM has strong robustness and can effectively deal with network fluctuations and partial information loss. For example, the temporary surge in demand from a single device may cause the immediate optimization algorithm to tilt most of the resources towards that user in exchange for short-term gains. Besides being unfair to other users, the short-term optimal solution will have a negative impact on the overall optimality and stability of the system. LSTM can appropriately reduce the impact of abnormal fluctuations and calculate more stable allocation plans.

B. MULTIDIMENSIONAL LSTM MODEL DEVELOPMENT

In order to maximize both queue stability and utility optimization, a fully connected neural network can be employed to input current network states and output evaluations for each allocation plan, as described in Eq. (30). However, while general neural networks can reduce computation difficult when outputting an allocation policy, they are limited by single dimensional data such as channels, storage, or computation performance. This can sacrifice detailed analysis needed to explore optimal solutions.

The proposed system faces two main challenges in outputting near-optimal solutions. Firstly, single dimensional data is inefficient for predicting overall network states. Although a hierarchical network model can coordinate all the outputs of one dimension's prediction network, it brings new challenges on how to conclude all types of data to compute an allocation policy. The spatial and temporal characteristics of the devices must be considered as inputs to increase accuracy. Secondly, timeliness of the allocation policy influences actual performance. Due to network congestion, equipment abnormalities, and equipment mobility, the current optimal solution may not be suitable for future trends. Therefore, policies calculated based on the current state may not apply for the next several slots. Hence, we propose speculating on the allocation policy applicable to the current period based on multi-dimensional historical data of the past.

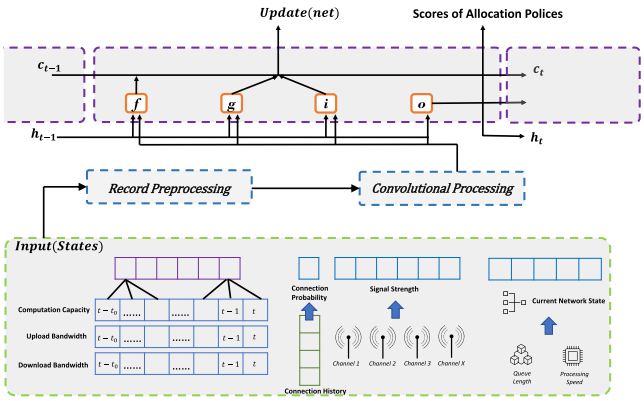


FIGURE 5. An LSTM-based time sequence processing schedule to calculate the allocation policy. The input data consists of a variety of network states, including vectors of communication, mobility, and computation capacity of local devices. The output is the evaluation values of the allocation policy.

Therefore, in this paper, we introduce a multi-dimensional LSTM neural network to overcome the above problems and output an optimized allocation policy to maximize the overall utility of Eq. (18).

Multi-dimensional LSTM is an effective neural network model for processing multi-dimensional time series data [26]. It gains popularity in various applications such as speech recognition, video content analysis, traffic prediction, and more. The model takes multi-dimensional input and calculates a combined prediction or evaluation result with respect to time. For instance, multi-dimensional LSTM can be used to predict hotel room reservations for the next period, thus assisting the hotel in preparing enough service resources. In this study, the states of the overall network devices are transformed into a time series sequence and used as input vectors for each time frame t, and the system outputs the candidate allocation policy with scores.

As shown in Fig. 6, the **time sequences are considered as input data, and the system outputs the detailed scores for each allocation plan.** The input vector is processed through the sequence folding layer, convolution layer, and sequence expansion layer to be explored the potential spatial correlation between devices [27], [28]. Then, it is restored to the vector sequence through the flat layer and fed to the classical LSTM layer for the analysis in the time dimension. At last, as shown in Fig. 5, the corresponding estimating value for each allocation plan is calculated through the final fully connected layer. Usually, the system chooses the allocation policy with the highest value to complete the resource management in the next period. We also design an online learning mechanism to store and update neural network weights to make the evaluation more precise and practical.

C. INPUT OF TIME SERIES VARIABLES

In this study, we consider a plane with coordinators where the federated learning system operates. Multiple base stations cover the entire plane, and mobile devices move along predefined traces. When the server publishes a specific FL task, the mobile device downloads the model data and completes the local training of the neural network. Subsequently, the devices immediately send the trained weights to the nearest base station upon completion.

The fusion process of the network weights is typically carried out synchronously or asynchronously. We assume that the fusion process is completed synchronously, starting after several devices have updated their training results. In addition to the unstable computational performance and energy limitations of the mobile devices themselves [29], various factors, such as signal-to-noise ratio, interference, multipath propagation, and the Doppler effect, can cause delays in task completion or failures. Therefore, our proposed system should consider these factors when assigning tasks.

When designing the input sequence of the training sample, we empirically consider the following factors as inputs.

- **Computation ability**: These factors are directly related to the efficiency of processing tasks. For example, we plan the available computing power of mobile devices, taking into account factors such as the power and frequency of device use, and assign a computation ability $C_{n,t}$ that changes over time.
- **Communication ability**: The two most important indicators are upload bandwidth and download bandwidth. In our scenario, we consider the impact of device movement on communication capabilities. The transmission of the general neural network model is completed within 1-10 seconds. We assume that the device's movement vector remains unchanged, and we take into account the influence of distance on the signal-to-noise ratio, converting all factors into actual bandwidth fluctuations.
- **Caching and serviceability**: These factors help determine the likelihood of a mobile device participating in training, considering factors such as caching ability, continuous access to the network time, signal strength, and remaining power, and are used to assign a reputation value to each device.

D. REWARD DESIGN BASED ON QUEUE DRIFT

The evaluation value represents the potential reward of an allocation policy and is aligned with the optimization goal. A higher reward value indicates that the current policy can achieve a higher utility of Eq. (18) compared to lower reward values.

However, the reward value of each policy only reflects the utility gain for the current time and state. As the utility processing ability of W_u in Eq. (20) changes with different tasks, the current allocation policies can have a significant impact on the future performance of the corresponding W_u. To estimate the total utility of the current allocation plan for a single task, we define the Lyapunov drift of the queue vector in a period of τ, where task k is allocated at time $\kappa \in [0, \tau]$.

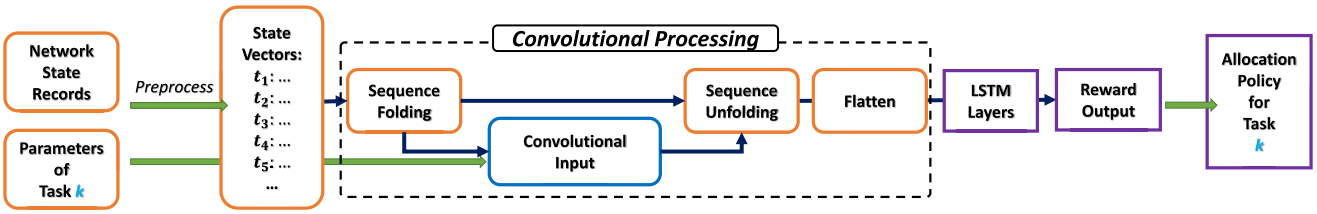


FIGURE 6. Utilize LSTM model with convolutional layers. Multi-dimensional vectors are used to calculate the data relevance of time and space through the pre-defined process of sequence expansion and convolution.

We have:

$$Qu(t+\tau)^2 - Qu(t)^2 \tag{31}$$
$$= Qu(t+\tau)^2 - Qu(t+\tau-1)^2 + Qu(t+\tau-1)^2$$
$$-Qu(t+\tau-2)^2 + \cdots + Qu(t+1)^2 - Qu(t)^2 \tag{32}$$
$$\rightarrow \underbrace{Qu(t)Au(t) - \sum_{\chi\in[\kappa,\tau]} Qu(t+\chi)Wu(t+\chi)}_{F_1}$$
$$+ \underbrace{\sum_{\chi\in[0,\kappa]} Qu(t+\chi)Wu(t+\chi)}_{F_2}. \tag{33}$$

The utility benefit gain results in a certain delay since the refunding is only confirmed when all devices complete the task k, and it is difficult to estimate the processing ability per time slot. To address this, we calculate the average utility gain based on Eq. (8) that affects each time slot. Next, we extract the allocation and processing operations part F_1 of the equation, which is related to the current allocation policy and future utility processing ability. Finally, we summarize the reward function R_n of a specific allocation policy on device n as follows:

$$R_n = \sum_{\chi\in[\kappa,\tau]} Qu(t+\chi)Wu(t+\chi) - Qu(t)Au(t) \tag{34}$$
$$-V(L(\tilde{Q}(t+\tau)) - L(\tilde{Q}(t))), \tag{35}$$

where V is a non-negative control parameter used to penalize deviation from the optimal average utility by at most $O(1/V)$ [30]. When τ tends towards infinity, the reward function calculates the overall drift caused by the task k. A higher value of R_n corresponds to a higher utility output, and it is more efficient for the queues to remain in Lyapunov stability.

The output of the reward function is a vector of $|N|$ dimensions, where each dimension represents the reward of assigning the task to a specific device $n \in N$. The system then outputs the current allocation policy based on the order of the utility value.

E. ALLOCATION POLICY UPDATING ALGORITHM

In this study, we utilize the proposed multi-dimensional LSTM model to output a specific allocation plan. Specifically, the optimization algorithm aims to calculate the corresponding allocation strategy for each time slot based on the utilization of network resources. The long-term algorithm is not intended to solve the optimal solution for the entire time frame T, but rather to solve the current optimal solution based on recent historical data and network status. As an online real-time updating algorithm, the methods we apply can solve the optimal solution for the entire time frame; however, the results obtained in this way may only have mathematical significance and may lack practical significance in real-world operations.

To generate the input data for the multi-dimensional LSTM, we convert the various factors discussed in the previous section into time series vectors. We then calculate the reward function of task assignment based on the current network states. Additionally, in the initial phase, the system attempts to assign tasks to all devices to obtain reward values, which are considered the initial training data set.

Inspired by the deep Q-learning mechanism, we utilize an online updating algorithm to adjust the output with changes in the network trend. As shown in the pseudocode of **Algorithm 20**, the LSTM learning model is updated effectively to match the current system capacity through the output of the new allocation policies.

Algorithm 20 is an online learning algorithm based on Q-learning that requires updates at each time slot t. However, since it still uses a convergence threshold ϵ to determine whether the while loop at each time slot should terminate, it is difficult to accurately determine the algorithm's complexity. To prevent long computation times that can result in slow convergence [31], we also set a limit on the number of learning rounds to help the algorithm stop in a reasonable time. The overall computation time still depends on the model size $\mathcal{W}$ and the states $\mathcal{S}$ that need to be updated.

If we only calculate the **FLOPs** (Floating Point Operations per Second) of a single neural network training, we can obtain an approximate computational complexity based on the structure in Fig. 6. For the initial multi-dimensional convolution part, assuming the sample number is S, the single sample input vector is C_i, the input sample type (computation ability, communication ability, etc.) is F, and the input time dimension is T_n, then the computational complexity of the input is $O(SC_iFT_n)$. To go through a three-layer convolutional network, assuming the maximum kernel size of the three convolutional layers is K, the kernel number is K', and the final approximation computational complexity is $O((2F +$

Algorithm 1 FL Service Oriented LSTM Based Queuing Optimization and Allocation Policy Calculation Algorithm

Require: Λ: Current Neural Network Model,
$\mathcal{W}$: Network Weight,
Π_0: Current Allocation Policy,
$\mathcal{S}$: Network State Field,
σ: Q-value Convergence Threshold,
υ: Learning Rate;
Ensure: Π: Allocation Policy $\Pi = \{\Pi_1, \Pi_2, \cdots\}$;
for each $t = 1; t \leq T; t++$; **do**
Start current state $\mathcal{S}_t \leftarrow \mathcal{S}_{t-1}$;
Start new network $\Lambda^+ \leftarrow \Lambda$ with $\mathcal{W}$;
while $\mathcal{W}^{k+1} - \mathcal{W} \geq \sigma$ **do**
Based on current network state $\mathcal{S}_k$ calculate next move by $\Lambda(\mathcal{S}_k)$;
Select the next state $\mathcal{S}_{k+1}$;
Match continuous domain $\mathcal{S}_k, \mathcal{S}_{k+1}$ to disc0rete domain $\hat{\mathcal{S}_k}, \hat{\mathcal{S}_{k+1}}$;
Save the allocation policy as $\hat{\mathcal{S}_k}, \hat{\mathcal{S}_{k+1}}, \Lambda(\mathcal{S}_k) \rightarrow \Pi$;
Update $\Lambda^+(\mathcal{S}_t) = R(\mathcal{S}_k) + \upsilon \max \Lambda^+(\mathcal{S}_{k+1})$;
Update new weight $\mathcal{W}$;
end while
Update $\Lambda \leftarrow \Lambda^+$ using $\mathcal{W}$;
Follow the policy Π_t calculated by $\Lambda(\mathcal{S}_t)$;
$t++$;
end for

$1)K^2K'C_iT_n$) after flattening. The input data dimension for LSTM is C_iT_n. Here we use a two-layer LSTM network, assuming the number of neurons is L, and the input data size at each time step is T_l, then the computational complexity after LSTM is $O(2T_n(4L^2+4LC_iT_l))$. Therefore, considering the case of training with S samples, the final computational complexity required for a single local update is $O((2F + 1)K^2K'C_iT_nS + 8(L^2 + LC_iT_l)T_nS)$.

VI. SIMULATION AND ANALYSIS

In this section, we evaluate the numerical performance of the proposed algorithm. Firstly, we provide details of the simulation platform used. Then, we compare our proposed methodology with different environment variables and algorithms to confirm its advantages.

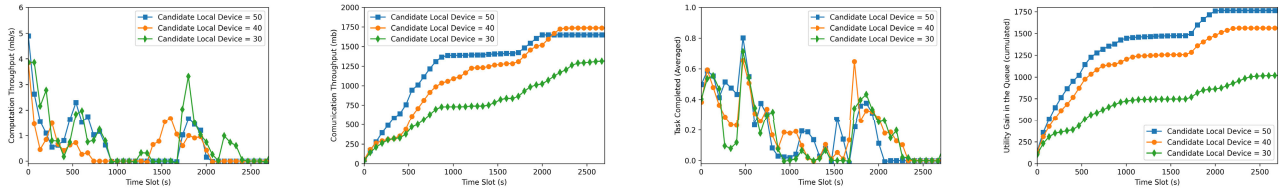
A. SIMULATION PLATFORM

This section outlines the simulation experiments conducted to validate our proposed approach. We simulate a virtual square of dimensions 10, 000m $\times$ 10, 000m for the mobile federated learning scenario. The distance between roads ranges from 50m to 200m, and mobile devices move along the streets at a rate of approximately 3m/s to 15m/s [32]. Base stations (BSs) are randomly distributed within the block separated by the roads. The communication mode uses the common 30 - 300 GHz frequency band [33], and the communication distance between mobile devices and the base station is maintained between 40m - 200m. The remote cloud server sends and receives FL tasks through the BS. Due to the instability of computing power and communication capacity, the cloud applies various algorithms to test the actual performance of the mobile-assisted FL system. We also preset 30 independent tasks in the candidate pool to simulate how multiple independent applications share public resources on cloud servers more efficiently. Considering privacy and competition, we do not consider the relation models that may exist between different learning tasks here. The introduction of the relationship model will make the convergence more efficient and make the cloud server achieve better resource utilization efficiency. But it still takes more work to evaluate the performance under strict convergence conditions.

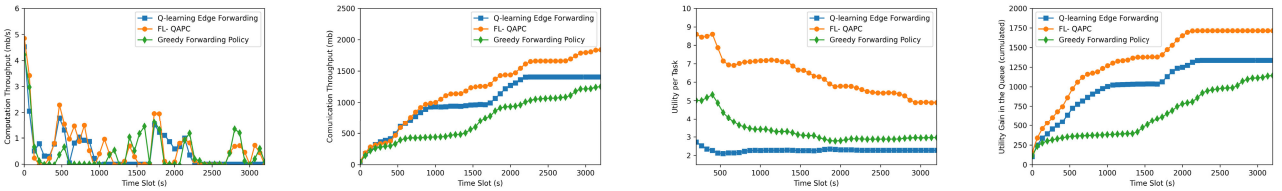
In addition to the proposed algorithm (**FL Service-oriented LSTM based Queuing Optimization and Allocation Policy Calculation Algorithm (FL-QAPC)**), we also evaluate two other algorithms for comparison. The first algorithm is based on the deep Q-learning framework [34] and is denoted as **Q-learning Edge Forwarding**. Using the same Lyapunov stability criterion in Eq. (34), this algorithm analyzes the correlation of queue vectors through multi-dimensional data input to achieve the optimized solution. It is based on the Deep Q-learning (DQL) algorithm and implements fine-grained resource allocation and management in Cloud Radio Access Network (C-RAN) based Network Function Virtualization (NFV). It can dynamically learn and optimize resource allocation to meet the needs of different users and applications, and improve network performance and efficiency. However, it does not have a predictive policy controller like the proposed algorithm, which makes it less efficient in terms of policy calculation. The second algorithm is the classical **Greedy Forwarding Policy**, which focuses on finding the optimal solution at each decision step based on the current state information and task. The algorithm aims to improve system performance and reliability by prioritizing tasks based on their utility. It achieves good computation efficiency and device reliability. However, it lacks the ability to perform holistic planning to handle multiple tasks in this scenario.

B. SYSTEM CAPACITY ANALYSIS

Firstly, we analyze the system's performance in terms of computation throughput and communication throughput when considering different candidate local devices. Fig. 7(a) illustrates the changes in actual computation throughput generated by varying numbers of devices. The x-axis represents time slots, which are considered in seconds. The y-axis represents the used computation throughput of the current time slot corresponding to the time axis. Here, we assume that the total global round of the tasks is the same. The faster the overall computation is completed, the sooner the task is considered complete. The blue line indicates that more devices participate and can be selected for local updating. Initially, more candidates participate in the early stage. The computation



(a) Computaion throughput which changes by different candidate devices. (b) Communication throughput which changes by different candidate devices. (c) Task complete frequence which changes by different candidate devices. (d) Overall utility gain which changes by different candidate devcies.



(e) Computaion throughput which changes by three algorithms. (f) Communication throughput which changes by three algorithms. (g) Task completed Ratio which changes by three algorithms. (h) Overall utility gain which changes by three algorithms.

FIGURE 7. FL system performance comparison using the different scenario settings and benchmarks.

throughput performs better than other stages during 0 - 505s. Although the performance of 40 candidates is relatively lower at first, it achieves computation queue stability faster than the result of 30 candidates.

Next, we evaluate other metrics, such as communication, completion rate, and utility gain. Fig. 7(b) illustrates the performance of the communication capacity used with different numbers of devices. Since the updating model size of a specific task is the same, earlier and higher communication resource utilization means faster completion of tasks. The proposed algorithm, represented by the blue line, quickly completes all rounds of updates. Fig. 7(c) shows the number of tasks completed in each time slot. Although the difference is not noticeable at first, more candidate devices help improve the rate of completing all tasks. For example, blue's 50 candidate solution manages more computing power and communication throughput. Thus, the performance of completing all FL tasks is better than others. Fig. 7(d) shows the utility gained after accomplishing the task update. From a long-term perspective, a higher utility shows the system keeps a more stable utility queue. However, the proposed algorithm with more candidates performs better than the other options. It completes the convergence rate faster, especially when the number of participating devices increases, and more critical tasks and devices that perform more efficiently can be intelligently selected.

C. ALGORITHM PERFORMANCE ANALYSIS

Next, we compare the computing performance of the three algorithms mentioned above. Fig. 7(e) shows the computation throughput of the algorithms. Since the proposed algorithm uses the LSTM model to predict possible future fluctuations based on historical records, it utilizes computing resources more efficiently. The second algorithm based on the deep Q-learning framework is not predictive of the future and focuses on the current state, resulting in a certain degree of delay compared to the proposed one. However, in the end, the performance of the two trends is the same due to the optimization mechanism of Lyapunov, as the global update is completed only after all the model's local updating is finished. According to the barrel principle, poorly performing devices eventually force the two algorithms to wait to complete a single device before being integrated globally. The greedy algorithm focuses on current utility gain and ignores the influence of global updating, which results in poorer performance compared to the other two algorithms.

Fig. 7(f) shows the performance of communication throughput. Due to the prediction mechanism of the LSTM model, the proposed algorithm infers the potential location of mobile devices for some time in the future based on their current locations, and predicts the actual communication capacity according to the coverage of the local base stations. As a result, the proposed algorithm outperforms the other two in terms of communication throughput. The blue line representing the second algorithm without real-time learning cannot cope with more random changes, resulting in lower communication throughput than the upper bound. The greedy algorithm only achieves local optimality and cannot guarantee performance in the long term.

Fig. 7(g) and Fig. 7(h) show the performance of the utility gain. The proposed algorithm consistently performs well in all stages of the y-axis. Our proposed algorithm outperforms the other two throughout the entire period. Based on the previous figures, we observe that the algorithm maximizes the utility gain of tasks by utilizing communication, computation, and global updating speed as much as possible. The benefit of the proposed Lyapunov stability control methodology is

that the algorithm achieves a better overall task completion rate compared to the Q-learning-based algorithm alone. However, the utility gain per slot is unexpectedly inferior to the greedy algorithm. We observe that the greedy algorithm can maximize the utility of the current single task at the expense of more available resources. In conclusion, in addition to optimizing the utilization of potential resources, our proposed algorithm performs better by considering an optimality trade-off between global updating and local updating.

VII. CONCLUSION

Coordinating remote resources to process federated learning tasks is one of the main challenges in cloud-supported multi-task distribution systems. To address the complexity of resource utilization, this research combines communication resources, computation ability, task convergence, and other factors in the edge-cloud FL system and integrates them into a single utility value as the optimization goal. To solve this optimization problem, we introduce Lyapunov drift to compress caching usage and maximize the system's task utility throughput. We design an online updating algorithm using the LSTM model to handle the calculation of the allocation policy in a time-varying environment. Our practical simulations demonstrate that our proposed algorithm performs significantly better than other benchmarks.

REFERENCES

[1] Z. Yan, J. Wu, G. Li, S. Li, and M. Guizani, "Deep neural backdoor in semi-supervised learning: Threats and countermeasures," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4827–4842, 2021.

[2] C. Zhang, M. Dong, and K. Ota, "Enabling computational intelligence for green Internet of Things: Data-driven adaptation in LPWA networking," *IEEE Comput. Intell. Mag.*, vol. 15, no. 1, pp. 32–43, Feb. 2020.

[3] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.

[4] C. Zhang, M. Dong, and K. Ota, "Heterogeneous mobile networking for lightweight UAV assisted emergency communication," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 3, pp. 1345–1356, Sep. 2021.

[5] L. U. Khan et al., "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 88–93, Oct. 2020.

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

[7] M. A. Bouras, F. Farha, and H. Ning, "Convergence of computing, communication, and caching in Internet of Things," *Intell. Converged Netw.*, vol. 1, no. 1, pp. 18–36, Jun. 2020.

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

[9] S. Huang, K. Ota, M. Dong, and F. Li, "MultiSpectralNet: Spectral clustering using deep neural network for multi-view data," *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 4, pp. 749–760, Aug. 2019.

[10] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-IID data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[11] Z. Hu, K. Shaloudegi, G. Zhang, and Y. Yu, "Federated learning meets multi-objective optimization," 2020, *arXiv:2006.11489*.

[12] X. Zhang, H. Gu, L. Fan, K. Chen, and Q. Yang, "No free lunch theorem for security and utility in federated learning," *ACM Trans. Intell. Syst. Technol.*, vol. 14, no. 1, pp. 1–35, Feb. 2023.

[13] T. T. Vu, H. Q. Ngo, M. N. Dao, D. T. Ngo, E. G. Larsson, and T. Le-Ngoc, "Energy-efficient massive MIMO for federated learning: Transmission designs and resource allocations," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 2329–2346, 2022.

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

[15] C. T. Dinh et al., "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021.

[16] D. Verma, S. Adhikari, and S. Ray, "Forwarding strategy in SDN-based content centric network," in *Proc. Int. Conf. Paradigms Commun., Comput. Data Sci.*, M. Dua, A. K. Jain, A. Yadav, N. Kumar, and P. Siarry, Eds. Singapore: Springer, Jan. 2022, pp. 49–62.

[17] X. Lin, J. Wu, J. Li, X. Zheng, and G. Li, "Friend-as-learner: Socially-driven trustworthy and efficient wireless federated edge learning," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 269–283, Jan. 2023.

[18] B. Dab, N. Aitsaadi, and R. Langar, "Joint optimization of offloading and resource allocation scheme for mobile edge computing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–7.

[19] W. Liu, X. Zang, Y. Li, and B. Vucetic, "Over-the-air computation systems: Optimization, analysis and scaling laws," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5488–5502, Aug. 2020.

[20] V. Smith, S. Forte, M. Chenxin, M. Takáč, M. I. Jordan, and M. Jaggi, "CoCoA: A general framework for communication-efficient distributed optimization," *J. Mach. Learn. Res.*, vol. 18, pp. 1–49, Jul. 2018.

[21] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.

[22] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3282–3299, Apr. 2020.

[23] Z. Jing, Q. Yang, Y. Wu, M. Qin, K. Sup Kwak, and X. Wang, "Adaptive cooperative task offloading for energy-efficient small cell MEC networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2022, pp. 292–297.

[24] H. Liu, Z. Hu, and Y. Song, "Lyapunov-based decentralized excitation control for global asymptotic stability and voltage regulation of multi-machine power systems," *IEEE Trans. Power Syst.*, vol. 27, no. 4, pp. 2262–2270, Nov. 2012.

[25] Q. He et al., "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.

[26] U. Yoshimura, T. Inoue, A. Tsuchiya, and K. Kishine, "Implementation of low-energy LSTM with parallel and pipelined algorithm in small-scale FPGA," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Jan. 2021, pp. 1–4.

[27] A. Azari, M. Ozger, and C. Cavdar, "Risk-aware resource allocation for URLLC: Challenges and strategies with machine learning," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 42–48, Mar. 2019.

[28] H. Shi and C. Wang, "LSTM-based traffic prediction in support of periodically light path reconfiguration in hybrid data center network," in *Proc. IEEE 4th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2018, pp. 1124–1128.

[29] C. Chen, Y.-H. Chiang, H. Lin, J. C. S. Lui, and Y. Ji, "Joint client selection and receive beamforming for over-the-air federated learning with energy harvesting," *IEEE Open J. Commun. Soc.*, vol. 4, pp. 1127–1140, 2023.

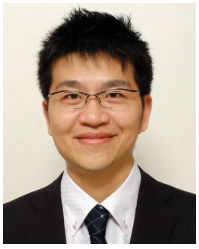
[30] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems* (Synthesis Lectures on Learning, Networks, and Algorithms), vol. 3, no. 1. San Rafael, CA, USA: Morgan & Claypool, Sep. 2010, pp. 1–211.

[31] C. Zhang, M. Dong, and K. Ota, "Employ AI to improve AI services: *Q*-learning based holistic traffic control for distributed co-inference in deep learning," *IEEE Trans. Serv. Comput.*, vol. 15, no. 2, pp. 627–639, Mar. 2022.

[32] Y. Wang, Z. Su, T. H. Luan, R. Li, and K. Zhang, "Federated learning with fair incentives and robust aggregation for UAV-aided crowdsensing," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3179–3196, Sep. 2022.

[33] D. Wang, B. Song, D. Chen, and X. Du, "Intelligent cognitive radio in 5G: AI-based hierarchical cognitive cellular networks," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 54–61, Jun. 2019.
[34] C. Zhang, M. Dong, and K. Ota, "Fine-grained management in 5G: DQL based intelligent resource allocation for network function virtualization in C-RAN," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 428–435, Jun. 2020.

CHAOFENG ZHANG (Member, IEEE) received the B.Eng. degree from Soochow University, China, in 2011, and the M.Eng. and Ph.D. degrees from the Muroran Institute of Technology, Japan, in 2016 and 2019, respectively. From March 2017 to April 2017, he was a Visiting Scholar with Soochow University, China. He is currently an Assistant Professor with the Advanced Institute of Industrial Technology (AIIT), Tokyo, Japan. His research interests include cloud computing, green IoT, wireless communication, and wireless positioning technology. He was a recipient of the IEEE VTS Tokyo Chapter 2016 Paper Award in 2016 and the Best Presentation Award in A3 Annual Workshop on Next Generation Internet and Network Security. He serves as an Associate Editor for *IEEJ Transactions on Electronics, Information and Systems* and *Frontiers in Space Technologies*.



MIANXIONG DONG (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science and engineering from The University of Aizu, Aizuwakamatsu, Japan. He was a JSPS Research Fellow with the School of Computer Science and Engineering, The University of Aizu, and a Visiting Scholar with the Broadband Communications Research (BBCR) Group, University of Waterloo, Waterloo, ON, Canada, supported by the JSPS Excellent Young Researcher Overseas Visit Program from April 2010 to August 2011. He is currently the Vice President and a Professor with the Muroran Institute of Technology, Muroran, Japan. He was selected as a Foreigner Research Fellow (a total of three recipients all over Japan) by Nippon Electric Company Computer and Communication (NEC C&C) Foundation in 2011. He is a Foreign Fellow of EAJ. He was a recipient of the 12th IEEE ComSoc Asia-Pacific Young Researcher Award 2017, the Funai Research Award 2018, the NISTEP Researcher 2018 (one of only 11 people in Japan) in recognition of significant contributions in science and technology, the Young Scientists' Award from MEXT in 2021, the SUEMATSU-Yasuharu Award from IEIEC in 2021, and the IEEE TCSC Middle Career Award in 2021. He is Clarivate Analytics in 2019, 2021, and 2022 Highly Cited Researcher (Web of Science).

KAORU OTA (Member, IEEE) was born in Aizuwakamatsu, Japan. She received the B.S. degree in computer science and engineering from The University of Aizu, Aizuwakamatsu, in 2006, the M.S. degree in computer science from Oklahoma State University, Stillwater, OK, USA, in 2008, and the Ph.D. degree in computer science and engineering from The University of Aizu in 2012. From March 2010 to March 2011, she was a Visiting Scholar with the University of Waterloo, Canada. From April 2012 to April 2013, she was a Japan Society of the Promotion of Science (JSPS) Research Fellow at Tohoku University, Japan. She is currently a Professor and a Ministry of Education, Culture, Sports, Science and Technology (MEXT) Excellent Young Researcher with the Department of Sciences and Informatics, Muroran Institute of Technology, Japan, where she is also the founding Director of the Center for Computer Science (CCS). She is selected as JST-PRESTO Researcher in 2021 and a fellow of EAJ in 2022. She was a recipient of the IEEE TCSC Early Career Award 2017, the 13th IEEE ComSoc Asia-Pacific Young Researcher Award 2018, the 2020 N2Women: Rising Stars in Computer Networking and Communications, the 2020 KDDI Foundation Encouragement Award, and the 2021 IEEE Sapporo Young Professionals Best Researcher Award, and the Young Scientists' Award from MEXT in 2023. She is Clarivate Analytics in 2019, 2021, and 2022 Highly Cited Researcher (Web of Science).