Game-Theoretic Design of Quality-Aware Incentive Mechanisms for Hierarchical Federated Learning

Gangqiang Hu, Jianmin Han, Jianfeng Lu, Juan Yu, *Member, IEEE*, Sheng Qiu, Hao Peng, Donglin Zhu, and Taiyong Li

***Abstract*—Hierarchical federated learning (HFL) improves the scalability and communication efficiency of the system and achieves load balancing at each level. Incentive mechanisms enhance participant motivation and optimize resource allocation for HFL. However, existing mechanisms mainly focus on maximizing individual utility from the quantity of client data while neglecting to optimize social utility from the learning quality perspective. Meanwhile, strategic behavior and heterogeneous devices can significantly degrade the performance of incentive mechanisms. To this end, we propose a quality-aware incentive mechanism (QAIM) for HFL to improve training efficiency. Specifically, we first systematically evaluate the learning quality of clients based on their training loss and historical records, which allows us to recruit high-quality clients for model updating selectively. Then, we model the cloud-edge-end interaction and cooperation as a three-layer Stackelberg game to analyze the strategies of participants and utilize carefully designed algorithms to derive the solution of the unique Stackelberg equilibrium (SE). Through the Pareto improvement of client association modeled as a coalition game, we can maximize social utility. Experimental results on both synthetic and real-world data sets demonstrate that our QAIM outperforms the state-of-the-art baselines, with an average increase in accuracy and social utility of 17% and 45%, respectively.**

***Index Terms*—Client association (CA), game theory, hierarchical federated learning (HFL), incentive mechanism, quality aware.**

I. Introduction

THE RAPID advancements in 5G and edge computing have catalyzed a significant expansion of the Internet of Things (IoT), resulting in an extensive collection of data from numerous IoT devices [1]. This data presents valuable opportunities for enhancing machine learning models and expanding the scope of automation and intelligence. Nonetheless, data extraction from IoT devices poses critical privacy challenges that must be addressed to responsibly leverage the potential of IoT within machine learning while maintaining ethical standards [2]. Federated learning (FL) is a promising technique for collaboratively training models with many devices while preserving privacy [3]. These devices can preserve privacy by avoiding uploading their raw data to the server. FL has shown impressive results in various IoT applications, such as healthcare, language processing, and security monitoring [4], [5], [6]. Consequently, it has garnered attention from both the academic and industrial communities. However, FL still faces several difficulties, such as communication bottlenecks and device dropouts [7]. Training may not be possible for devices located in geographically distant areas. These will result in potential negative impacts on the model's ability to generalize. To this end, researchers have proposed a new learning framework called hierarchical FL (HFL) [8], which has a cloud-edge-end structure. This framework allows end devices (clients) to upload their local model to an edge server, such as a base station, for edge model aggregation. A cloud server then aggregates the global model based on the intermediate parameters uploaded by the edge servers. HFL significantly increases network bandwidth and alleviates communication bottlenecks compared to cloud-based FL. By reducing expensive communication through efficient end-edge updates, the execution duration and the number of local iterations can be significantly decreased [9].

Previous work on HFL focused mainly on improving training performance by proposing novel learning or resource allocation algorithms [7], [10], [11], [12]. Due to the significant costs involved in model training, such as computation and communication expenses, self-interested participants are unlikely to have any motivation to take part in HFL training. Meanwhile, the quality of learning and client association (CA) vary for each end device, so failure to effectively engage high-quality devices in training can affect the accuracy of the final model. Therefore, designing an effective incentive mechanism to encourage their participation in training models is essential. However, most existing research about incentive mechanisms focused on traditional cloud-based FL [13], [14], [15]. The researchers studied designing incentives between a cloud server and multiple clients using game theory, contracts, and auctions. These studies have

Manuscript received 5 February 2024; revised 27 March 2024; accepted 12 April 2024. Date of publication 26 April 2024; date of current version 25 July 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62372343 and Grant 62072411; in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LR21F020001; and in part by the Key Research and Development Program of Hubei Province under Grant 2023BEB024. *(Corresponding author: Jianmin Han.)*

Gangqiang Hu, Jianmin Han, Juan Yu, Sheng Qiu, Hao Peng, and Donglin Zhu are with the School of Computer Science and Technology, Zhejiang Normal University, Jinhua 321004, China (e-mail: hugangqiang@zjnu.edu.cn; hanjm@zjnu.cn; 1147899155@qq.com; yujuan@zjnu.edu.cn; hpeng@zjnu.cn; zdl0730@163.com).

Jianfeng Lu is with the School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430065, China (e-mail: lujianfeng@wust.edu.cn).

Taiyong Li is with the School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu 611130, China (e-mail: litaiyong@gmail.com).

Digital Object Identifier 10.1109/JIOT.2024.3394170

achieved some success, but they cannot be applied directly to HFL. A few incentive mechanisms are geared toward HFL, but they always ignore the impact of low-quality clients and CA on the model [16], [17].

Considering the aforementioned difficulties and gaps, developing a quality-aware incentive mechanism (QAIM) for HFL becomes critically important. Three challenges hinder the design of QAIM for HFL.

C1—Heterogeneous Devices: Due to the characteristics of HFL itself, heterogeneity is prevalent in the HFL process. Different end devices and edge servers have different quantities of data, and the cost of collecting and processing each data may also vary. These heterogeneities make it difficult to accurately measure the cost and contribution of each participant and the corresponding rewards in designing incentives [18]. Therefore, how can we effectively incentivize end devices and edge servers with diverse individual properties to participate in a heterogeneous scenario?

C2—Strategic Behavior: In HFL, all participants are inherently selfish and rational. Driven by their interests, they are motivated to maximize their utility. This is a multifaceted optimization problem that considers both costs and benefits [19]. Then, how can we attain the desired incentive effects and motivate all participants to engage in a self-interested and rational manner?

C3—Unreliable Participants: The quality of training data is critical to model training. The performance of end devices can vary significantly during the training process. Model convergence problems may result from aggregating too many low-quality model updates. However, recruiting high-quality devices to participate in the HFL is challenging and requires the development of carefully designed evaluation criteria for quality [20]. Therefore, how can we design an effective quality evaluation criterion without wasting excessive computational resources?

Inspired by game-theoretic insights, we employ the Stackelberg and coalitional game methodology to design and implement incentives for HFL to analyze the strategies of participants. Stackelberg games with leaders and followers are suitable for hierarchical structures, while coalitional games are also suitable for CAs, such as edge server selection by clients. The complex relationship between each participant, which is both competitive and cooperative, can be better explained and analyzed from a game perspective. Given the inherent challenges of FL in accurately predicting the quality of the final model, we address it by transforming the model quality maximization problem into utility maximization. Similar transformations converting model quality to a utility function exist in many existing articles [21], [22]. We use three concave functions to define the utility of the participants, which is consistent with objective laws, and the game equilibrium can be efficiently analyzed. The utility function not only calculates the benefits for each participant, but also estimates the final model quality. This mechanism enhances the credibility and performance of each participant, encouraging positive behavior and discouraging negative behavior.

The key contributions of this article are outlined as follows.

1) We propose a novel QAIM for HFL using game theory. The proposed mechanism translates the training progress in cloud-edge-end systems into subgames interconnected by utility functions, which depict strategic responses at each layer.
2) We transform the model quality maximization problem into a utility maximization problem and use training loss and history to evaluate the learning quality of each end device as a way to incentivize more high-quality devices to contribute. Moreover, we model the cloud-edge-end cooperation as a three-layer Stackelberg game and the CA as a coalition game. The Stackelberg equilibrium (SE) and closed-form solutions are derived to guide participants in obtaining the desired outcomes effectively. Social utility is maximized by applying a coalitional game, where CAs are iteratively optimized as continuous Pareto improvements.
3) Experiments on synthetic and real-world data sets (MNIST, EMNIST, CIFAR10, and SVHN) demonstrate that the proposed mechanism improves on average 17% and 45% in terms of test accuracy and social utility compared to the state-of-the-art.

The remainder of this article is organized as follows. In Section II, we describe some related work. In Section III, we focus on modeling and formulation of the problem. Next, Section IV analyzes and designs the incentive mechanism. In Section V, we conduct experiments on the mechanism we designed. Finally, Section VI summarizes our work and future research directions.

II. Related Works

Cloud-based FL typically requires high-communication costs and network bandwidth requirements, while edge-based FL often faces the inability to handle complex computational and administrative tasks with the limited capacity of edge servers [23]. To this end, literature [8] proposed HFL with a cloud-edge-end structure. Compared with traditional FL, HFL partially aggregates at the edge level, which can effectively reduce communication bottlenecks and has better scalability to scale to many devices. Most of the existing work [24], [25] focused on improving the accuracy and convergence of the model by designing efficient HFL algorithms and optimizing resource allocation and client selection. However, these are optimistic estimates of each participant's voluntary and unpaid participation in the training. Given this impracticality, investigating a potent incentive mechanism becomes crucial.

Incentive mechanisms in FL or HFL have been extensively studied to encourage participants to contribute more training data and computational resources. For example, incentive mechanisms based on Stackelberg games can be designed by having a central authority or platform owner as the leader, who sets the rules or incentives, while the participants act as followers. Literature [26] proposed an incentive mechanism in a continuous and dynamic environment, utilizing a two-stage Stackelberg game framework to maximize each participant's utility. Literature [27] used the Stackelberg game approach to design a multifactor incentive mechanism for FL, modeled the interaction between the task publisher and the data owner as a two-stage game, and derived the optimal equilibrium solution. In [17], the incentive mechanism designed for HFL maximizes the utility of each participant using a three-layer Stackelberg game approach. These papers are designed solely to maximize the utility of each participant without considering the social utility and data quality. Incentive mechanisms based on contract theory often involve a principal and multiple agents, and the principal often designs an optimal contract to maximize the model owner's utility [14]. Literature [28] designed a hierarchical incentive mechanism by designing an optimal contracting mechanism for each model owner and a coalition formation algorithm (CFA) to maximize the benefits for each model owner. Literature [29] utilized contract theory to develop an efficient incentive mechanism that encourages mobile devices with high-quality data to engage in FL. However, these incentives based on contract theory tend to maximize only the utility of the model owner while ignoring the utility of the client. Incentive mechanisms based on auction theory can be applied to allocate resources, such as computation power or data, among participants in a system [30]. Literature [31] designed a QAIM that leverages differential privacy to safeguard local models and true costs from untrustworthy parameter servers. Furthermore, they developed a multidimensional reverse auction mechanism to incentivize high-quality and low-cost data owners to engage in FL. Although this article devises incentives for quality awareness, it is not directly applicable to HFL. In addition, literature [32] designed quality-aware incentives for HFL based on a matching game, but this is modeled only from the perspective of resource allocation and does not fully consider the characteristics of participants' selfishness and rationality. Literature [16] proved and solved the Nash equilibrium (NE) by designing an incentive mechanism for HFL based on blockchain, taking into account the multidimensional properties of the participants. However, this article did not consider the social utility or the impact of client quality on the model.

To overcome the shortcomings of the status quo, we propose a QAIM for HFL based on game theory. This mechanism effectively addresses the challenges of unreliable participants and heterogeneous, selfish behavior during model training.

III. System Model and Problem Formulation

This section presents the basic framework for network modeling and the steps involved in the system and incentive processes. Then, we provide a utility function for each participant and formalize the optimal solution to the given problem.

A. System Overview

A typical network model for HFL includes a cloud server, multiple edge servers (e.g., base stations), and many end devices, as shown in Fig. 1. For the rest of this article, we will always use the client instead of the end device.

1) *Cloud Servers:* The training task is deployed on the cloud server to achieve optimal model performance and corresponding utility. In this context, the cloud server incentivizes the edge servers through rewards, indirectly encouraging the clients to engage in the model training process.

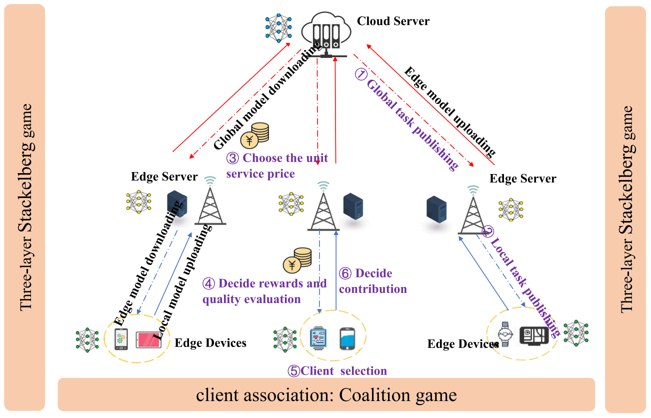


Fig. 1. Framework of our QAIM for HFL.

2) *Edge Servers:* Multiple edge servers make up the collection $\mathcal{M} = \{1, 2, \ldots, m, \ldots, M\}$, where a coalition of clients S_m is associated with each edge server $m \in \mathcal{M}$. We assume that each client can choose which edge server to serve and that it does not change during training once selected. As a pivotal element within HFL, edge servers assess each client's learning quality. Furthermore, they aggregate model parameters and ensure an equitable distribution of rewards between the cloud and the clients.
3) *End Devices (Clients):* There are N clients forming a collection $\mathcal{N} = \{1, 2, \ldots, n, \ldots, N\} = \cup_{m=1}^{M} S_m$. If a client n serves an edge server m, we set $\mu_n = m$. We have $n \in S_{\mu_n}$ and denote $\mathbf{S}_n = S_{\mu_n} \subseteq \mathcal{N}$. Each client n has a data set $\mathcal{X}_n$ of learning quality θ_n, where the training loss and the history record can evaluate the value of θ_n. At each round of training, the client n selects the amount of data x_n to participate in the training based on its resource cost.

Learning Process in HFL: Every client seeks to collaboratively train the global model ω utilizing its local data set. The global model ω is derived by minimizing the total empirical risk across all aggregated local data sets, represented by the loss function $F(\omega)$. After each client performs τ local updates, it uploads its trained model to the edge server for edge aggregation. After σ times of edge model aggregation, the edge server uploads the edge model to the cloud server for global aggregation. The local model $\omega_n(t)$ is referred to as the model after the tth local update, which changes as follows [16]:

$$\omega_n(t) = \begin{cases} \omega_n(t-1) - \eta_t \nabla F_n(\omega_n(t-1)), & t \nmid \tau \\ \frac{\sum_{n\in \mathbf{S}_n} x_n[\omega_n(t-1) - \eta_t \nabla F_n(\omega_n(t-1))]}{\sum_{n\in \mathbf{S}_n} x_n}, & t \mid \tau, t \nmid \tau\sigma \\ \frac{\sum_{n\in \mathcal{N}} x_n[\omega_n(t-1) - \eta_t \nabla F_n(\omega_n(t-1))]}{\sum_{n\in \mathcal{N}} x_n}, & t \mid \tau\sigma. \end{cases} \quad (1)$$

The incentive process in HFL is as follows.

1) *Step ①:* The cloud server sends the global training task to the edge server.

TABLE I
NOTATION SUMMARY FOR THIS ARTICLE

Variable	Description
μ_n, x_n, π_n	Edge server number to which client n belongs, Amount of data contributed by client n, Strategies for Client n
M, N	Number of edge server, Number of clients
θ_n, q_n, ξ	Quality of Learning Assessments for Client n, Model quality of client n, Thresholds for learning assessment quality
f_n, c_n, ζ	Client's CPU cycle frequency, Number of CPU cycles required to train a unit of data, Effective capacitance parameter of the computing chipse
C_n^E, E_n	Unit cost of data collection by client n, Energy consumption for client n training
p_n, ρ	Proportion of contribution from client n, Exponential degradation parameters
$c^S\left(c_m^{cmp}, c_m^{com}, S_m\right)$	Cost of edge servers m
$J_n, \omega_n(t)$	The unit training cost of client n, Client n's local model at round t
α_m, β_m	Risk aversion parameter , Reward scaling coefficient
λ, τ, σ	Weighting parameter, Local update round, Edge aggregation round
$\lvert\mathcal{X}_n\rvert, \mathcal{N}, \mathcal{M}$	Client n's dataset, Client set, Edge server set
S_m, Π	Client set for edge server m, Partition of coalitions
γ_m, P	Rewards for Edge Server m, Unit Service Pricing for the cloud server

2) *Step ②:* Each edge server breaks down the task and sends it to the clients, assuming that each end device is willing to participate in the training.
3) *Step ③:* After all the edge servers and clients are aware of the training task, the cloud server can decide the per-unit service price to be sent to the edge servers through a service price decision.
4) *Step ④:* Depending on the price of the unit service, each edge server can decide rewards plan to be paid to clients and estimate the learning quality of each client based on historical learning records.
5) *Step ⑤:* Each client needs to decide which edge server to serve.
6) *Step ⑥:* Once the clients have established coalitions based on their situations and strategic considerations, they will decide the amount of data to contribute based on their different learning qualities and costs.

After these steps, they follow the HFL training process described before for model training. The process of cloud-edge-end cooperation is modeled as a three-layer Stackelberg game, and each client chooses an edge server and forms a coalition, which is modeled as a coalition game. The important notations are listed in Table I.

The reward strategy of each edge server m is $\gamma_m \in \mathcal{R}$. The contribution plan of each client n consists of choosing which edge server to serve μ_n and the amount of data x_n to be provisioned, which together constitute the contribution plan $\pi_n = (\mu_n, x_n)$. Also, we let $\pi_{-n} = (\mu_{-n}, x_{-n})$ denote the set of all strategies except π_n. If client n does not want to participate in training the model, it will set $x_n = 0$. The payoff obtained by a client n depends on the model quality contribution $q_n = \theta_n x_n$ of other clients (i.e., $q_{-n} = (q_i \quad \forall i \in \mathbf{S}_n \backslash \{n\})$) in the same coalition $\mathbf{S}_n$, and is also related to the total amount of payoffs γ_{μ_n} that the corresponding edge server can provide. Each edge server provides compensation based on a percentage p_n of the client's contribution. Obviously, there are $\sum_{n \in \mathbf{S}_n} p_n = 1$. Similar to the previous work [13], [28], we have

$$p_n = q_n \Bigg/ \left(q_n + \sum_{i \in \mathbf{S}_n \backslash \{n\}} q_i \right). \tag{2}$$

Therefore, the payoff of client n can be expressed as follows:

$$A(q_n, q_{-n}) = p_n \gamma_{\mu_n} \tag{3}$$

where the $A(\cdot)$ function is a concave function and nondecreasing, showing a marginal decreasing effect in evaluating the edge server's contribution to the client. From this equation, it is clear that having a higher quality client will lead to a higher percentage of revenue share.

The edge server evaluates the learning quality of the client for each round. The most accurate approach would be to run each local model on all test data sets to calculate the test accuracy. However, implementing this approach would result in substantial additional costs and resources. The acquisition of training loss values does not require additional overhead, as it is generated directly during the training process. Thus, we measure the learning quality using the amount of training loss reduced during each iteration. Let us consider that a specific iteration round t starts at time t_s and finishes at time t_e. Clients must submit their local model updates within the specified timeframe $[t_s, t_e]$, or the edge server will refuse their submissions.

The average loss values at t_s and t_e are $\text{loss}(t_s)$ and $\text{loss}(t_e)$, respectively. Inspired by [20], the quality of the training data for the client n in iteration t is defined as

$$\theta = \text{loss}(t_s) - \text{loss}(t_e). \tag{4}$$

Estimation of Learning Quality: After quantifying the quality of learning, we subsequently assess each client's current learning quality. Specifically, in the context of iterative distributed learning, we utilize the participant's historical quality records in learning tasks to estimate the quality of its model updates. Assuming that the client n has trained the model, we can use the historical quality record to estimate the quality of the contribution of the current iteration t. The number of rounds and quality of previous iterations can be expressed as $t_0, t_1, \ldots, t_r$ and $(\theta_n^{t_0}, \theta_n^{t_1}, \ldots, \theta_n^{t_r})$. The learning quality of individual clients may change over time, and it is intuitively understood that more recent quality records hold greater significance than older records. Instead of assigning equal weight to all quality records, we employ a weighting system based on their freshness [20]. An exponential forgetting function allocates weights, prioritizing recent quality records while devaluing outdated ones. Therefore, the evaluation quality of client n before iteration t can be expressed as

$$\widehat{\theta}_n^t = \frac{\sum_{k=0}^{r} \rho^{t_r - t_k} \theta_n^{t_k}}{\sum_{k=0}^{r} \rho^{t_r - t_k}}. \tag{5}$$

Later in this article, we use θ_n instead of $\widehat{\theta}_n^t$ to simplify the presentation. We use $q_n = \theta_n x_n$ to denote the model quality of the client n.

The client's expenses primarily consist of computational and collection expenses, directly correlated with the volume of data used for training purposes. The variable C_n^E is the cost to the client for the acquisition of a single unit of data. f_n, c_n, and ζ denote the client's CPU cycle frequency, the number of CPU cycles required to train a unit of data, and the effective capacitance parameter of the computing chipset. Therefore, the energy consumption for a single local iteration can be expressed as follows [33]:

$$E_n(x_n) = \zeta c_n x_n f_n^2. \tag{6}$$

The representation of the training cost can be derived under the assumption that the individual characteristics of the participants remain constant throughout a single round

$$\text{cost}_n^E = \kappa E_n(x_n)\tau\sigma + C_n^E x_n. \tag{7}$$

In the above equation, κ is a weighting parameter for energy relative to training needs. For the sake of simplicity, we can define the training cost for client n as $\text{cost}_n^E = J_n x_n$, where J_n represents the cost per unit.

B. Problem Formulation

We will make a formal representation of the utility of each client, edge server, and cloud server. Although our initial goal is to optimize the model quality of each participant (i.e., clients, edge servers, cloud servers), the final model quality is always challenging to predict. Inspired by [13], we convert the problem of maximizing the quality of the model to maximizing the utility while considering the effect of cost. The utility function of the client n is formulated as follows:

$$\begin{cases} \arg\max\limits_{q_n} \quad U_n(q_n, q_{-n}) \quad \forall n \in S_m, m \in [1, M] \\ \text{s.t.} \begin{cases} U_n(q_n, q_{-n}) = A(q_n, q_{-n}) - J_n x_n \\ q_n = \theta_n x_n \\ 0 \le x_n \le |\mathcal{X}_n| \\ \theta_n \ge \xi \end{cases} \end{cases} \tag{8}$$

where $|\cdot|$ denotes the cardinality of a set. For each client n, the quality of the learning assessment must meet or exceed the threshold ξ to participate in the HFL.

The cloud announces a unit service price P for motivation before HFL training starts. For an edge server m, it has to decide a reward program γ_m for its corresponding clients. Its goal is to make itself available to better edge models by optimizing rewards to obtain higher utility. The edge server's utility function is directly related to the quality of the local models provided by the clients. The higher the quality of the local models, the greater the utility of the edge server. Therefore, we optimize the reward design so clients can provide higher quality model updates. Edge servers have a corresponding computational and communication overhead $c^S(c_m^{cmp}, c_m^{com}, S_m)$. c_m^{cmp} and c_m^{com} are unit computational costs and communication costs. Based on a risk-averse model like in [34] and [35], we can formally define the utility of edge server m as

$$\begin{cases} \arg\max\limits_{\gamma_m} \quad U_m(\gamma_m), m \in [1, M] \\ \text{s.t.} \begin{cases} U_m(\gamma_m) = \text{In}\left(\alpha_m + \sum_{n \in S_m} q_n P\right) - \frac{\gamma_m}{\beta_m} \\ -\sigma \cdot c^S\left(c_m^{cmp}, c_m^{com}, S_m\right) \\ \gamma_m \ge 0. \end{cases} \end{cases} \tag{9}$$

The edge server determines the risk aversion parameter α_m to align with its costs, while the reward scaling coefficient β_m is employed. The cost of an edge server is linearly proportional to the number of its corresponding clients, so we define the cost of an edge service as $c^S(c_m^{cmp}, c_m^{com}, S_m) = (c_m^{cmp} + c_m^{com})|S_m|$.

The goal of the server, as the model owner, is to improve the quality of the global model by incentivizing edge servers and clients to provide higher quality model updates. The cost consumption is the per unit service price P. To analyze the relationship between the learning quality of clients and the expected performance of the trained model, we define that the model quality can be viewed as a concave function with respect to all of the learning quality provided by clients [13]. Therefore, the benefit of the cloud service can be defined as $\lambda \cdot g(\sum_{n \in \mathcal{N}} q_n)$, where $\lambda > 0$ is greater than 0 and is a parameter set in advance. $g(\cdot)$ is a concave function on the quality of the model update. The higher the quality, the higher the value of $g(\cdot)$, but it also has a decreasing marginal effect. Thus, the utility of the server can be formalized as follows:

$$\begin{cases} \arg\max\limits_{P} \quad U_C(P) \\ \text{s.t.} \begin{cases} U_C(P) = \lambda \cdot g\left(\sum_{n \in \mathcal{N}} q_n\right) - \sum_{n \in \mathcal{N}} q_n P \\ P \ge 0. \end{cases} \end{cases} \tag{10}$$

In QAIM, we design payment rules $\mathbb{F}$, reward rules $\mathbb{C}$, service pricing (SP) rules $\mathbb{A}$, and CA rules $\mathbb{Z}$.

Definition 1: QAIM for HFL can be described as a $(\mathbb{F}, \mathbb{C}, \mathbb{A}, \mathbb{Z})$.

1) $\mathbb{F}$ represents the payment rule used to determine the payments for clients in the HFL system, i.e.,

$$A(q_n, q_{-n}) = \frac{q_n}{\left(q_n + \sum_{i \in \mathbf{S}_n \setminus \{n\}} q_i\right)} \gamma_{\mu_n}. \tag{11}$$

2) $\mathbb{C}$ represents the reward rule that defines how should each edge server set the amount of compensation reward γ_m as a way to determine its utility, i.e.,

$$\gamma_m \leftarrow \arg\max_{\gamma_m \ge 0} \quad U_m \quad \forall m = 1, \ldots, M. \tag{12}$$

3) $\mathbb{A}$ represents the SP rule that determines the pricing strategy for the services provided by the HFL system, i.e.,

$$P \leftarrow \arg\max_{P} \quad U_C. \tag{13}$$

4) $\mathbb{Z}$ represents the CA rule, that is, which edge server each client chooses to serve, i.e.,

$$\Pi \leftarrow N \times N \times \cdots \times N. \tag{14}$$

According to QAIM, the payoff received by a client n is related to the quality of the local model q_n. Thus, it is in each client's interest to actively participate in the training mission and to be incentivized so that each client can contribute at a high level. At the same time, there is a strong correlation between the compensation paid by edge servers, the number of clients they can attract, and the total local model quality. The price per unit of service posted by the cloud server directly impacts the participation of each edge server. It indirectly affects the local model quality contribution of the clients. All clients, edge servers, and cloud servers want to maximize their utility. QAIM has positive incentives for clients, edge servers, and cloud servers and aims to maximize social utility. The total social utility is the sum of the utility of the client, the edge server, and the cloud server. Thus, the design problem of QAIM can be formalized as follows

Definition 2: The QAIM design problem is formulated as

$$\begin{cases} \max\limits_{(\mathbb{F},\mathbb{C},\mathbb{A},\mathbb{Z})} \mathcal{U} = \sum_{n\in\mathcal{N}} U_n + \sum_{m\in\mathcal{M}} U_m + U_C \\ \qquad n \in [1, N], m \in [1, M] \\ \text{s.t.} \begin{cases} U_n = \max\limits_{0\le q_n\le\theta_n|\mathcal{X}_n|,\theta_n\ge\xi} A(q_n, q_{-n}) - J_n x_n \\ U_m = \max\limits_{\gamma_m\ge 0} \text{In}\left(\alpha_m + \sum_{n\in[1,N]} q_n P\right) - \frac{\gamma_m}{\beta_m} \\ -\sigma \cdot c^S\left(c_m^{cmp}, c_m^{com}, S_m\right) \\ U_C = \max\limits_{P\ge 0} \lambda \cdot g\left(\sum_{i\in\mathcal{N}} q_i\right) - \sum_{i\in\mathcal{N}} q_i P. \end{cases} \end{cases} \tag{15}$$

IV. Design of QAIM

This section aims to analyze the existence of an optimal solution to the design problem of the QAIM and explore the methods to identify it. To maximize social utility and encourage active participation in HFL training, we investigate the incentive problem from a game-theoretic perspective.

We begin by modeling the three-layer Stackelberg game to solve for the NE solution in which each participant maximizes their utility individually. Since these equilibrium solutions are not necessarily optimal, we construct a coalitional model to maximize social utility by changing the CAs. The three-layers Stackelberg game contains three subgames, e.g., the data training (DT) game, the reward plan (RP) game and the SP game. We will solve these three game problems using the backward induction method [36]. First, we compute the DT game, i.e., the amount of training data contributed by each client. Then, we compute the RP game, i.e., how much reward each edge server needs to give the clients. Finally, we compute the SP game, i.e., the price per unit of service that the cloud server should provide to the edge server. For each subgame, we derive their corresponding NE. Meanwhile, the NE of the DT, RP, and SP games together form the SE, defined as follows.

Definition 3 (SE): Let P^*, $\Gamma^* = (\gamma_1^*, \ldots, \gamma_M^*)$ and $X^* = \{x_1^*, \ldots, x_N^*\}$ be NE of the SP, RP and DT game, respectively, $(P^*, .\Gamma^*, X^*)$ is an SE for the three-layer Stackelberg game if

$$\begin{cases} U_C(P^*, \Gamma^*, X^*) \ge U_C(P, \Gamma^*, X^*) \\ U_m\left(P^*, \gamma_m^*, \gamma_{-m}^*, X^*\right) \ge U_m\left(P^*, \gamma_m, \gamma_{-m}^*, X^*\right) \\ \forall m \in [1, M] \\ U_n\left(P^*, \Gamma^*, x_n^*, x_{-n}^*\right) \ge U_n\left(P^*, \Gamma^*, x_n, x_{-n}^*\right) \quad \forall n \in [1, N]. \end{cases} \tag{16}$$

A. Analysis of the Three-Layer Stackelberg Game

The equilibrium strategies for the followers in this Stackelberg game can be characterized as the strategies that effectively respond to the leader's offers optimally. Following the backward induction solution [37], we first derive the DT game, then the RP game, and finally the SP game.

In the DT game, the edge server is the leader, and the clients are the followers. Given the others' strategies x_{-n}, for each client n, its goal is to maximize its own utility function $U_n(x_n, x_{-n})$. To optimize the strategy for each client, we calculate the first and second derivatives of utility

$$\frac{\partial U_n(x_n, x_{-n})}{\partial x_n} = \frac{\sum_{k\in S_m\setminus\{n\}} \theta_k x_k}{\left(\sum_{k\in S_m} \theta_k x_k\right)^2} \cdot \gamma_m - J_n \tag{17}$$

and

$$\frac{\partial^2 U_n(x_n, x_{-n})}{\partial {x_n}^2} = -\frac{2\gamma_m \sum_{k\in S_m\setminus\{n\}} \theta_k x_k}{\left(\sum_{k\in S_m} \theta_k x_k\right)^3} < 0. \tag{18}$$

Let (17) equals 0, and we can derive the optimal response strategy for client n as

$$x_n^*(x_{-n}) = \begin{cases} 0, & \gamma_m < J_n \cdot \sum_{i\in S_m\setminus\{n\}} \theta_i x_i \; or \; \theta_n < \xi \\ \frac{\sqrt{\frac{\gamma_m \cdot \sum_{i\in S_m\setminus\{n\}} \theta_i x_i}{J_n}} - \sum_{i\in S_m\setminus\{n\}} \theta_i x_i}{\theta_n}, & \text{otherwise.} \end{cases} \tag{19}$$

Lemma 1: The DT game has a unique NE for a given value of γ_m and x_{-n}.

Proof: See Appendix A. ■

Corollary 1: For clients that meet the learning assessment quality conditions $\theta_n \ge \xi$ in the coalition S_m, we can select them according to their unit cost from the lowest to the highest. For each new client joining the coalition, the following conditions are met:

$$J_n < \frac{\sum_{i\in S_m} J_i}{|S_m| - 1} \tag{20}$$

and

$$x_n^*(x_{-n}) = \begin{cases} 0, & n \notin S_m \\ \frac{\frac{\gamma_m(|S_m|-1)}{\sum_{i\in S_m} J_i}\left(1 - \frac{J_n(|S_m|-1)}{\sum_{i\in S_m} J_i}\right)}{\theta_n}, & n \in S_m. \end{cases} \tag{21}$$

Proof: See Appendix B. ■

In the RP game, the cloud server is the leader, and the edge servers are the followers. Depending on the optimal response strategy of each client, the edge server needs to decide γ_m to optimize its utility $U_m, m \in [1, M]$. We can rewrite the best strategy x_n^* as

$$x_n^* = Y_n \gamma_m / \theta_n \tag{22}$$

where

$$Y_n = \frac{|S_m| - 1}{\sum_{i\in S_m} J_i}\left(1 - \frac{J_n(|S_m| - 1)}{\sum_{i\in S_m} J_i}\right). \tag{23}$$

To analyze the utility function $U_m(\gamma_m)$ for edge server m, we need to compute its first-order and second-order derivatives with respect to γ_m. We have

$$\frac{\partial U_m(\gamma_m)}{\partial \gamma_m} = \frac{\sum_{n\in S_m} Y_n P}{\alpha_m + \gamma_m \sum_{n\in S_m} Y_n P} - \frac{1}{\beta_m} \tag{24}$$

and

$$\frac{\partial^2 U_m(\gamma_m)}{\partial \gamma_m{}^2} = -\frac{\left(\sum_{n\in S_m} Y_n P\right)^2}{\left(\alpha_m + \gamma_m \sum_{n\in S_m} Y_n P\right)^2} < 0. \quad (25)$$

By making the equation of the first order derivative zero, we can calculate the optimal reward strategy γ_m^* for the edge server

$$\gamma_m^* = \beta_m - \frac{\alpha_m}{\sum_{n\in S_m} Y_n P}. \quad (26)$$

Lemma 2: Given a service unit price P, there is a unique NE in the RP game.

Proof: See Appendix C. ■

In the SP game, the primary goal of the cloud, assuming a leadership role, is to optimize its utility through the unit service strategy P^*. We still perform the derivation as follows:

$$\frac{\partial U_C(P)}{\partial P} = \lambda \cdot g'\left(\sum_{n\in\mathcal{N}} \beta_m Y_n - \frac{\sum_{m=1}^{M} \alpha_m}{P}\right) - \sum_{n\in\mathcal{N}} \beta_m Y_n \quad (27)$$

and

$$\frac{\partial^2 U_C(P)}{\partial P^2} = \lambda \cdot g''\left(\sum_{n\in\mathcal{N}} \beta_m Y_n - \frac{\sum_{m=1}^{M} \alpha_m}{P}\right). \quad (28)$$

By making the equation of the first-order derivative equal to 0, we can calculate the optimal unit service price strategy P^* for the cloud server.

For example, in the case that $g(\cdot) = 1/2\,\mathrm{In}(1+x)$, the optimal strategy for the cloud server can be derived as follows:

$$P^* = \sqrt{\frac{0.5\lambda \sum_{m=1}^{M} \alpha_m}{\Delta(1+\Delta)} + \frac{\left(\sum_{m=1}^{M} \alpha_m\right)^2}{4(1+\Delta)^2}} + \frac{\sum_{m=1}^{M} \alpha_m}{2(1+\Delta)} \quad (29)$$

where $\Delta = \sum_{n\in\mathcal{N}} \beta_{\mu_n} Y_n$.

Lemma 3: Given the parameters of the client and edge services and the values of α and β, there is a unique NE in the SP game.

Proof: See Appendix D. ■

Theorem 1: From Lemmas 1–3, we have a unique SE.

B. Analysis of the Client Association

The three-layer Stackelberg game only solves the equilibrium solution for each participant but does not guarantee an optimal solution for social utility. Therefore, we optimize the final social utility by improving the CA game modeled by the coalition game. The CA game will be analyzed as follows.

Assuming that there are N HFL clients $\mathcal{N} = \{1, 2, \ldots, N\}$, coalition S_m represents a group of HFL clients (can also be called clients in the coalitional game) interested in participating in the same set of HFL [38].

Definition 4 (Coalition Partition): There is a coalition partition Π where the set $\Pi = \{S_1, S_2, \ldots, S_m, \ldots, S_M\}$ is defined, which is comprised of the HFL clients. Here, for $m = 1, \ldots, M$, every $S_m \subseteq \mathcal{N}$ is a disjoint coalition such that $\cup_{m=1}^{M} S_m = \mathcal{N}$ and $S_j \cap S_k = \emptyset$ for $j \neq k$.

To give an example of a coalition partition, we have a set of seven clients $\mathcal{N} = \{1, 2, 3, 4, 5, 6, 7\}$, and then three edge servers with a current CA $S_1 = \{1, 2\}, S_2 = \{3, 4\}, S_3 = \{5, 6, 7\}$. Therefore, this is a coalition partition $\Pi = \{S_1, S_2, S_3\}$.

Lemma 4 (Nonsuperadditivity): The grand coalition is not always formed through coalitional games. In contrast, separate small disjoint coalitions will emerge finally.

Proof: See Appendix E. ■

Definition 5 (Switch Rule): We propose two coalition formation rules. They are listed below.

1) *Split Rule:* When we have a coalition S, we can split this coalition into two smaller coalitions S' and S'', i.e., $S \rightarrow S' \cup S''$.
2) *Joining Rule:* When there is a coalition S with a client n in it, it wants to leave the current coalition S and join a new coalition S', i.e., $S \cup S' \rightarrow (S \backslash \{n\}) \cup (S' \cap \{n\})$.

Definition 6 (Coalition Altruistic Preference Rule): We define the partial order relationship for the entire coalition partition $\succ$. Such a change in coalition structure is allowed only when the partial order relationship of overall social utility improvement is satisfied [39].

The partition is changed from Π_i to Π_{i+1} according to the splitting rule. For example, Π_i contains the coalition S, and Π_{i+1} contains S' and S'' after the splitting of s. If $v(\Pi_i) > v(\Pi_{i+1})$, then the coalitional altruistic preference rule is satisfied. A preference relationship is defined to establish the coalition formation process. Each client can select and assess two distinct coalitions based on the preference criteria in this situation.

We have designed a CFA to maximize social utility, as outlined in Algorithm 1. First, we get the necessary parameters before running the algorithm. In line 1, we randomly initialize the division of a coalition. We then do an iterative update of the algorithm from line 2 to line 18 until the variable ConIteration becomes False. For each iteration, we first make the variable ConIteration False. From line 4 to line 10, we make a swift rule that randomly selects a coalition S and then splits it into two subcoalitions $S'$and S''. We perform this switch operation if the social utility is higher after the shift. Similarly, we perform another swift rule operation from line 11 to line 17.

Theorem 2 (Convergence): Starting with a randomly constructed coalition state Π_0, according to our proposed switch rule, it must eventually converge. The time complexity of Algorithm 1 does not exceed $O(K(N^2 + M^2)$, where K is the Bell number [40].

Proof: See Appendix F. ■

V. Experiments Results

This section will evaluate our proposed incentive mechanism on synthetic and real-world data sets. Our proposed method is evaluated using PyTorch 2.0.0, a well-known machine-learning framework. Initially, we will outline the fundamental setup of the experiment and establish the control benchmarks. Then, we will show the results of our comparison experiment.

A. Experiments Setting

Model and Setting: Our training model employs the convolutional neural network (CNN). The batch size is set to 32. The

Algorithm 1: CFA to Maximize the Social Utility

```
Input: $\mathcal{N} = \{1, 2, \ldots, N\}, \mathcal{M} = \{1, 2, \ldots, M\}, \alpha, \beta, \lambda, C_n^E, c_n, f_n, \zeta, \xi, \tau, \sigma, c^S(c_m^{cmp}, c_m^{com})$;
Output: The final partition $\Pi_f = \{S_1, S_2, \ldots, S_M\}$, the final social utility and the solution of $x_n, \gamma_m, P$;
1  $\Pi_0 \leftarrow$ random produce a partition satisfy coalition condition;
2  repeat
3      ConIteration $\leftarrow$ False;
4      $S \leftarrow$ Randomly select a coalition from $\Pi_c$;
5      $S', S'' \leftarrow$ divide coalition $S$ into two separate coalitions meeting the coalition condition;
6      Calculate $x_n, \gamma_m, P$ and the utility of $v(\Pi_c)$ and $v(\Pi_{c+1})$ according to Eq. (19), Eq. (26), Eq. (29);
7      if $v(\Pi_{c+1}) > v(\Pi_c)$ then
8          $\Pi_{c+1} \leftarrow \Pi_c \backslash \{S\} \cup \{S'\} \cup \{S''\}$;
9          ConIteration $\leftarrow$ True;
10     end
11     $S, S' \leftarrow$ Randomly select two coalitions from $\Pi_c$;
12     $n \leftarrow$ Randomly select a client from $S$;
13     Calculate $x_n, \gamma_m, P$ and utility of $v(\Pi_c)$ and $v(\Pi_{c+1})$ according to Eq. (19), Eq. (26), Eq. (29);
14     if $v(\Pi_{c+1}) > v(\Pi_c)$ then
15         $\Pi_{c+1} \leftarrow \Pi_c \backslash \{S\} \backslash \{S'\} \cup \{\{S' \backslash \{n\}\} \cup \{S'' \cup \{n\}\}$;
16         ConIteration $\leftarrow$ True;
17     end
18 until ConIteration=False;
```

maximum number of global aggregations is 100. Furthermore, the learning rate, represented as η, is 0.01, and the stochastic gradient descent (SGD) momentum is 0.05.

Data Sets: This experiment uses a synthetic data set and four real-world data sets, which are well-known data sets (MNIST [41], EMNIST [42], CIFAR10 [43], SVHN [44]) for machine learning classification. In the case of the synthetic data set, the specific parameters of the experimental setup are listed in Table II. In the case of real-world data sets, MNIST data set comprises handwritten digits. EMNIST comprises six distinct partitions, with the largest being EMNIST ByClass, encompassing 814 255 characters distributed across 62 imbalanced classes. The CIFAR10 data set comprises 50 000 training images and 10 000 test images distributed across 10 distinct classes. The SVHN data set comprises real-world house number images comprising 73 000 training and 26 000 test data samples.

Baselines: To verify the effectiveness of our QAIM, we compared our mechanisms with the following three typical mechanisms.

1) *FLQA:* Hui et al. [32] proposed a QAIM for HFL using a matching game, where social utility through CAs is not considered here.
2) *FLSG:* Zhao et al. [17] proposed an incentive mechanism for HFL based on the Stackbelberg game that does not consider the client training quality and maximization of social utility.

TABLE II
EXPERIMENTAL PARAMETERS

Parameters	Values
Number of edge server M	5
Number of clients N	15
Unit cost of data collection C_n^E by client n	$[0.05, 0.1]$
Client's CPU cycle frequency f_n	$[1, 2]$
Effective capacitance parameter of the computing chipset ζ	1
Cost of edge servers $c^S\left(c_m^{cmp}, c_m^{com}, S_m\right)$	$[0.1, 0.2]$
Number of CPU cycles required to train a unit of data c_n	$[1, 2]$
Risk aversion parameter α_m	$[1, 3]$
Reward scaling coefficient β_m	$[1, 2]$
Weighting parameter λ	5
Local update round τ	2
Edge aggregation round σ	3
Thresholds for learning assessment quality ξ	0.4

3) *RS:* RS is a random mechanism, defined as each client randomly selecting their contribution level. For example, the clients randomly choose their data volume contribution from 0 to $|\mathcal{X}_n|$. That is to say, there is no incentive for the clients in RS. In addition, all other parameter settings are the same as those proposed in this article for QAIM.

Metrics: We employ two metrics to evaluate and compare different mechanisms using synthetic data sets. The first is social utility, which is the metric to evaluate the performance of our mechanisms. The second is the utility of the cloud server, a metric that is used to assess the utility as a task initiator. The higher the utility of the cloud server, the higher the quality of the cloud server's model. We employ two metrics to evaluate and compare different mechanisms using real-world data sets. The first is prediction accuracy, which is usually used to measure how accurate the predictions of a trained model are on a test set. The second metric is the training loss, quantifying the training error incurred during the model training process. For all data sets, we separately consider both with and without low-quality participants.

B. Results on Synthetic Data Set

Fig. 2 shows the corresponding changes in social and cloud server utility as the number of participating clients and edge servers vary when there are no low-quality clients. It is evident that with an increase in the number of clients and edge servers, there is a concurrent increase in both the social and utility of the cloud servers. This is because the more participants there are, the more resources the system can utilize and the greater the utility will naturally be. On average, our QAIM improves social utility by 6.9% and cloud server utility by 7.5% relative to the other three mechanisms. We find very little difference in the performance of the other three mechanisms. This is because while FLQA considers quality, it operates under the assumption that participating clients exhibit high quality, thereby reducing the distinction between FLQA and FLSG. Moreover, despite the absence of client incentives, RS enhances client matching and holds a comparative edge over FLQA and FLSG in this regard. In summary, all three baseline models exhibit lower efficacy levels when compared with our proposed QAIM.

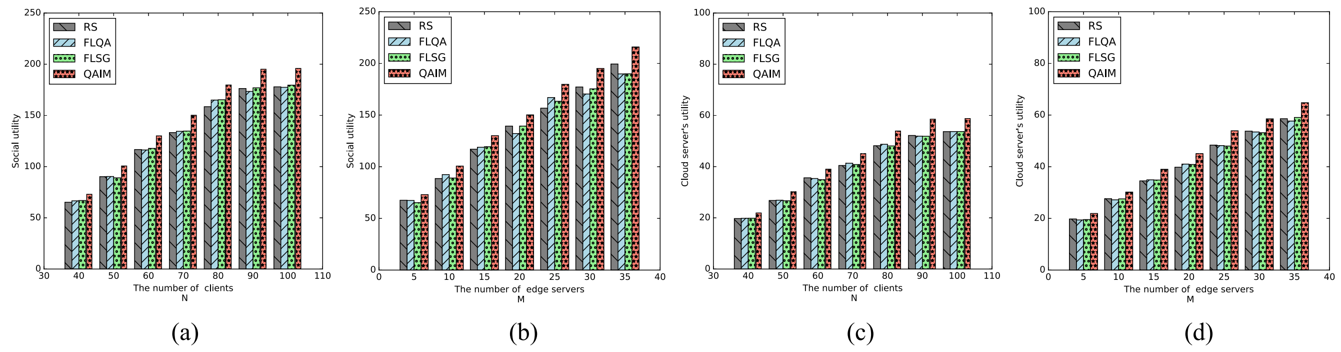


Fig. 2. Social utility and cloud server's utility of the compared mechanisms against *N* and *M* under the synthetic data set without low-quality clients. (a) Social utility. (b) Social utility. (c) Cloud server's utility. (d) Cloud server's utility.

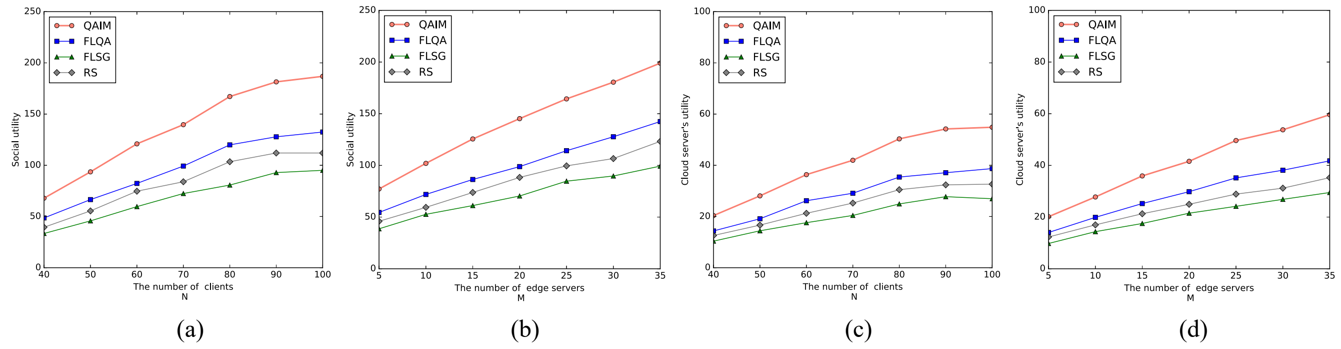


Fig. 3. Social utility and cloud server's utility of the compared mechanisms against *N* and *M* under the synthetic data set with 30% low-quality clients. (a) Social utility. (b) Social utility. (c) Cloud server's utility. (d) Cloud server's utility.

Fig. 3 shows the corresponding changes in social utility and cloud server utility as the number of participating clients and edge servers vary when there are 30% low-quality clients with error labels. It is evident that with an increase in the number of clients and edge servers, there is a concurrent increase in both the social and utility of the cloud servers. This is because the more participants there are, the more resources the system can utilize and the greater the utility will naturally be. On average, our QAIM improves social utility by 45% and cloud server utility by 42% relative to the other three mechanisms. This is because QAIM considers the quality of the client's model and maximizes the social utility through CAs. FLQA is a close second because it considers client-side model quality. RS considers client-side associations but ignores incentives for clients, so it is further behind. Neither model quality nor CAs are considered in FLSG, so it is the least useful.

C. Results on Real-World Data Set

Fig. 4 shows the changes in test accuracy and training loss for different mechanisms under four real-world data sets as the number of communication rounds increases. Although the lines are a bit dense, it can still be seen that our QAIM has a faster convergence rate, higher test accuracy, and lower test loss than the other three mechanisms when there are no low-quality clients. The CIFAR10 and SVHN data sets are more difficult to train than the MNIST and EMNIST data sets, so overall, they have lower accuracy and higher loss. Both the MNIST and EMNIST data sets have close to 98% test accuracy, while CIFAR10 and SVHN only have up to 59% and 87.2% test accuracy. This is because our QAIM enhances the final model quality of the cloud server by improving CAs and maximizing social utility. FLQA and FLSG are roughly in the middle of the pack because they boost participation levels in model training by incentivizing participants at each level. The least effective is RS because it does not incentivize better client participation in model training.

Fig. 5 shows how all the mechanisms compare with each other under four different real-world data sets when the client has low quality with the wrong data labels. We categorize the percentage of clients possessing incorrect data labels into three cases: 1) 10%; 2) 30%; and 3) 50%. These mislabels are not present initially but gradually become more numerous as training progresses. For each global training round performed, the mislabeling rate is an increase of 1%. Our QAIM can better incentivize high-quality clients to participate in training, in this case, clients with erroneous data sets, while low-quality clients can hardly participate in training. In Fig. 5(a) and (b), our mechanism enables to achieve more than 90% test accuracy even in the case of clients with 50% low quality. In Fig. 5(c) and (d), although the test accuracy is reduced compared to the case when there are no mislabeled clients, a very high-test accuracy of about 54% in CIFAR10 and about 78% in SVHN is still achieved relative to other mechanisms. On average, our QAIM improves test accuracy by about 17% in the presence of 30% low-quality clients. This is because our proposed incentive mechanism is able to update the data quality of each client in each global training round, thus motivating

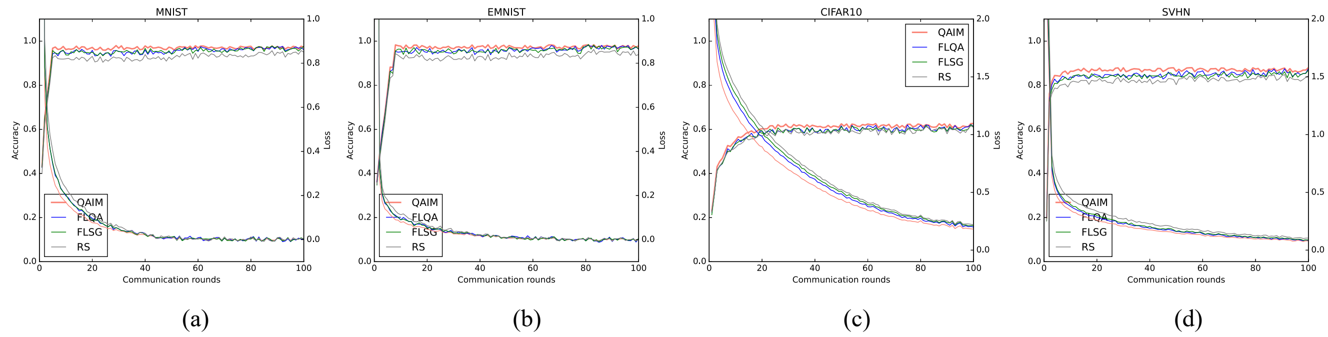


Fig. 4. Prediction accuracy and training loss without low-quality clients. (a) MNIST. (b) EMNIST. (c) CIFAR10. (d) SVHN.

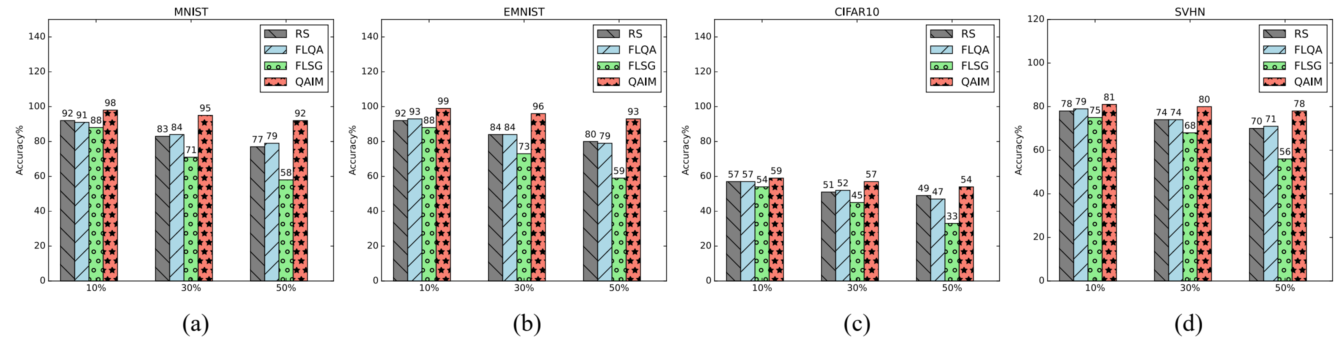


Fig. 5. Prediction accuracy with 10%, 30%, and 50% low-quality clients. (a) MNIST. (b) EMNIST. (c) CIFAR10. (d) SVHN.

high-quality clients to participate in training. FLQA and RS have about the same accuracy because they both take into account the model quality of the client and filter the client by model quality. FLSG does not consider the effect of client quality on the model at all.

VI. Conclusion

In this article, the QAIM for HFL based on game theory, aiming to improve the model training quality of HFL substantially has been proposed. Specifically, we have devised and executed three technical constituents: 1) estimation of learning quality; 2) modeling of game theory; and 3) optimization of social utility. Extensive experimentation has been conducted on five data sets, yielding results that prove the effectiveness of our mechanism in terms of participant incentive. For future work, we will investigate how to incentivize each participant to participate in training and improve model quality when each participant has computational and communication constraints and budgetary constraints in large scale FL.

Appendix A
Proof of Lemma 1

According to [13], a unique NE solution exists when and only when the following two conditions are satisfied: 1) the strategy sets are convex, bounded, and closed, and 2) the utility functions in the strategy space are quasi-concave and continuous.

Since for client n, its strategy space is to decide how much data quantity to contribute to participate in the training. The interval is $[0, |\mathcal{X}_n|]$, So it is clearly convex, bounded, closed, and continuous. According to the second derivative of the utility function U_n of client n, we get

$$\frac{\partial^2 U_n(x_n, x_{-n})}{\partial {x_n}^2} = -\frac{2\gamma_m \sum_{k\in S_m\setminus\{n\}} \theta_k x_k}{\left(\sum_{k\in S_m} \theta_k x_k\right)^3} < 0. \tag{30}$$

Thus, the function is quasi-concave. So Theorem 1 is proved.

Appendix B
Proof of Corollary 1

First, we take the first-order derivative of the utility function U_n with respect to q_n for client n. We get

$$\frac{\gamma_m}{\sum_{i\in S_m} q_i} - \frac{\gamma_m q_n}{\left(\sum_{i\in S_m} q_i\right)^2} - J_n. \tag{31}$$

Then, we let the first-order equation above become 0. The derivation yields

$$(|S_m| - 1)\gamma_m = \sum_{i\in S_m} q_i \sum_{i\in S_m} J_i. \tag{32}$$

Next, we get an expression for the sum of all q_n

$$\sum_{i\in S_m} q_i = \frac{(|S_m| - 1)\gamma_m}{\sum_{i\in S_m} J_i}. \tag{33}$$

Combining (32) and (33), we get

$$\sum_{i\in S_m\setminus\{n\}} q_i^{ne} = \frac{(|S_m| - 1)^2 \gamma_m J_n}{\left(\sum_{i\in S_m} J_i\right)^2}. \tag{34}$$

Since the condition q_n is greater than or equal to must be satisfied, we get it by bringing the above equation into this condition

$$J_n < \frac{\sum_{i\in S_m} J_i}{|S_m|-1}. \tag{35}$$

We bring (34) into (8), and we get

$$\begin{aligned} q_n^* &= \sqrt{\frac{\gamma_m \cdot \sum_{i\in S_m\setminus\{n\}} q_i}{J_n}} - \sum_{i\in S_m\setminus\{n\}} q_i \\ &= \frac{\gamma_m(|S_m|-1)}{\sum_{i\in S_m} J_i} - \frac{J_n\gamma_m(|S_m|-1)^2}{\left(\sum_{i\in S_m} J_i\right)^2} \\ &= \frac{\gamma_m(|S_m|-1)}{\sum_{i\in S_m} J_i}\left(1-\frac{J_n(|S_m|-1)}{\sum_{i\in S_m} J_i}\right). \end{aligned} \tag{36}$$

Since $q_n^* = \theta_n x_n^*$, we end up with an expression for x_n^* as

$$x_n^* = \begin{cases} 0, & n \notin S_m \\ \dfrac{\frac{\gamma_m(|S_m|-1)}{\sum_{i\in S_m} J_i}\left(1-\frac{J_n(|S_m|-1)}{\sum_{i\in S_m} J_i}\right)}{\theta_n}, & n \in S_m. \end{cases} \tag{37}$$

Appendix C
Proof of Lemma 2

We can follow the proof of Lemma 1.

For each edge server, their strategy space is $[0,\infty]$. Therefore, this set of strategies is clearly convex and continuous. Although theoretically, the value of γ can be infinite, in fact, the value of γ cannot be infinite due to cost constraints and the diminishing marginal utility of the utility function. Therefore, this set of strategies is necessarily bounded and closed as well.

Next, we prove the quasi-concave property of the utility function. We obtain by second-order derivation

$$\frac{\partial^2 U_m(\gamma_m)}{\partial \gamma_m{}^2} = -\frac{\left(\sum_{n\in S_m} Y_n P\right)^2}{\left(\alpha_m + \gamma_m \sum_{n\in S_m} Y_n P\right)^2} < 0. \tag{38}$$

Thus, the utility function is quasi-concave, so a unique NE exists.

Appendix D
Proof of Lemma 3

Similar to the proof of Lemma 1, the previously mentioned convexity, continuity, boundedness, and closure of strategy spaces can be easily proved but are omitted here. Next, we focus on proving the proposed quasi-concave.

We have previously obtained the second-order derivative of the cloud server utility function as

$$\frac{\partial^2 U_C(P)}{\partial P^2} = \lambda \cdot g''\left(\sum_{n\in\mathcal{N}} \beta_m Y_n - \frac{\sum_{m=1}^M \alpha_m}{P}\right). \tag{39}$$

The g function we set up is a log function, a distinctly concave function. Inside the parameter, $\sum_{n\in\mathcal{N}} \beta_m Y_n - (\sum_{m=1}^M \alpha_m)/P$ is an obvious concave function because everything else is fixed and only P is a variable. Moreover, $-(1/P)$ is a distinctly concave function. So, we have to prove that their composite function is concave.

The composite function $g(f)$ represents the application of function f to the input first, followed by the application of function g to the result. In other words, for a given input x, we first calculate $f(x)$, and then the function g will take the result of $f(x)$ as input. Since g is a concave function, it implies that for any x_1 and x_2, as well as any $t \in [0,1]$.

The inequality satisfies

$$g(tx_1 + (1-t)x_2) \le tg(x_1) + (1-t)g(x_2). \tag{40}$$

On the other hand, the inequality satisfies

$$f(tx_1 + (1-t)x_2) \le tf(x_1) + (1-t)f(x_2). \tag{41}$$

By combining the above inequalities, we can derive

$$\begin{aligned} g(f(tx_1 + (1-t)x_2)) &\le g(tf(x_1) + (1-t)f(x_2)) \\ &\le tg(f(x_1)) + (1-t)g(f(x_2)). \end{aligned} \tag{42}$$

This indicates that $g(f(x))$ is a concave function, as the inequality

$$g(f(tx_1 + (1-t)x_2)) \le tg(f(x_1)) + (1-t)g(f(x_2)) \tag{43}$$

when x_1, x_2 and t are all between 0 and 1.

Therefore, since g and f are all concave functions, their composite function $g(f)$ is also concave. This proves that there is a unique NE for the cloud server.

Appendix E
Proof of Lemma 4

Because of the theory in [38] and [45], The nonsuperadditivity of the coalitional game has been first proven. We assume here that there exist two disjoint coalitions, S_1 and S_2. They have two different options. One is to exist disjointly, and the other is to form the grand coalition $S_1 \cup S_2$. A coalitional game is referred to as superadditive if the condition $v(S_1 \cup S_2) \ge v(S_1) + v(S_2)$ is met. Here, $v(S)$ represents the benefit that a coalition S can obtain. However, (9) shows that the marginal contribution decreases as $|S|$ (the number of clients) in the coalition increases. This further implies that the coalition game is not superadditive $v(S_1 \cup S_2) < v(S_1) + v(S_2)$.

Appendix F
Proof of Theorem 2

The formation of the coalition is comparable to a series of transfer operations. Following the rules of the coalitional game, each current state Π_c can be transferred to the next state Π_{c+1}, and the Pareto improvement in social utility must be satisfied each time for the transfer to take place. From the initial state Π_0, our algorithm will produce the next transition [46]

$$\Pi_0 \to \Pi_1 \to \cdots \to \Pi_c \to \Pi_{c+1} \tag{44}$$

where the implementation of a shift procedure is indicated by the $\to$ symbol. Every application of the shift rule generates two possible cases.

As we know, the number of coalitions a client can join is limited and cannot exceed the Bell number limit [40]. Hence, it is inevitable that the transformation sequence will reach termination, culminating in convergence to a specific partition, denoted by Π_f.

References

[1] R. Huo et al., "A comprehensive survey on blockchain in Industrial Internet of Things: Motivations, research progresses, and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 88–122, 1st Quart., 2022.

[2] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5476–5497, Apr. 2021.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2016, pp. 1–10.

[4] X. Zhu, S. Wen, S. A. Çamtepe, and Y. Xiang, "Fuzzing: A survey for roadmap," *ACM Comput. Surveys*, vol. 54, nos. 11S, pp. 1–36, 2022.

[5] J. Lu, H. Liu, Z. Zhang, J. Wang, S. K. Goudos, and S. Wan, "Toward fairness-aware time-sensitive asynchronous federated learning for critical energy infrastructure," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3462–3472, May 2022.

[6] 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./Oct. 2022.

[7] W. Y. B. Lim et al., "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 536–550, Mar. 2022.

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

[9] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6535–6548, Oct. 2020.

[10] B. Xu, W. Xia, W. Wen, P. Liu, H. Zhao, and H. Zhu, "Adaptive hierarchical federated learning over wireless networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 2, pp. 2070–2083, Feb. 2022.

[11] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2020, pp. 1–9.

[12] W. Y. B. Lim, J. S. Ng, Z. Xiong, D. Niyato, C. Miao, and D. I. Kim, "Dynamic edge association and resource allocation in self-organizing hierarchical federated learning networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3640–3653, Dec. 2021.

[13] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, vol. 7, pp. 6360–6368, Jul. 2020.

[14] N. Ding, Z. Fang, and J. Huang, "Incentive mechanism design for federated learning with multi-dimensional private information," in *Proc. Int. Symp. Model. Optim. Mobile, Ad-Hoc Wireless Netw.*, 2020, pp. 1–8.

[15] L. Li, X. Yu, X. Cai, X. He, and Y. Liu, "Contract-theory-based incentive mechanism for federated learning in health crowdsensing," *IEEE Internet Things J.*, vol. 10, no. 5, pp. 4475–4489, Mar. 2023.

[16] X. Wang, Y. Zhao, C. Qiu, Z. Liu, J. Nie, and V. C. M. Leung, "InFEDge: A blockchain-based incentive mechanism in hierarchical federated learning for end-edge-cloud communications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3325–3342, Dec. 2022.

[17] Y. Zhao, Z. Liu, C. Qiu, X. Wang, F. R. Yu, and V. C. M. Leung, "An incentive mechanism for big data trading in end-edge-cloud hierarchical federated learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2021, pp. 1–6.

[18] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Mach. Learn. Res.*, vol. 139, 2021, pp. 12878–12889.

[19] Y. Yuan, L. Jiao, K. Zhu, and L. Zhang, "Incentivizing federated learning under long-term energy constraint via online randomized auctions," *IEEE Trans. Wireless Commun.*, vol. 21, no. 7, pp. 5129–5144, Jul. 2022.

[20] Y. Deng, F. Lyu, J. Ren, Y.-C. Chen, P. Yang, Y. Zhou, and Y. Zhang, "Fair: Quality-aware federated learning with precise user incentive and model aggregation," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[21] G. Huang, X. Chen, O. Tao, Q. Ma, L. Chen, and J. Zhang, "Collaboration in participant-centric federated learning: A game-theoretical perspective," *IEEE Trans. Mobile Comput.*, vol. 22, no. 11, pp. 6311–6326, Nov. 2023.

[22] J. Lu, H. Liu, R. Jia, J. Wang, L. Sun, and S. Wan, "Toward personalized federated learning via group collaboration in IIoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 8, pp. 8923–8932, Aug. 2023.

[23] P. Singh, G. S. Gaba, A. Kaur, M. Hedabou, and A. Gurtov, "Dew-cloud-based hierarchical federated learning for intrusion detection in IoMT," *IEEE J. Biomed. Health Inf.*, vol. 27, no. 2, pp. 722–731, Feb. 2023.

[24] J. Wang, S. Wang, R.-R. Chen, and M. Ji, "Local averaging helps: Hierarchical federated learning and convergence analysis," 2020, *arXiv:2010.12998*.

[25] B. Xu, W. Xia, J. Zhang, X. Sun, and H. Zhu, "Dynamic client association for energy-aware hierarchical federated learning," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2021, pp. 1–6.

[26] G. Xiao, M. Xiao, G. Gao, S. Zhang, H. Zhao, and X. Zou, "Incentive mechanism design for federated learning: A two-stage Stackelberg game approach," in *Proc. IEEE 26th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, 2020, pp. 148–155.

[27] Y. Chen, H. Zhou, T. Li, J. Li, and H. Zhou, "Multifactor incentive mechanism for federated learning in IoT: A Stackelberg game approach," *IEEE Internet Things J.*, vol. 10, no. 24, pp. 21595–21606, Dec. 2023.

[28] W. Y. B. Lim et al., "Hierarchical incentive mechanism design for federated machine learning in mobile networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9575–9588, Oct. 2020.

[29] J. Kang, Z. Xiong, D. T. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim, "Incentive design for efficient federated learning in mobile networks: A contract theory approach," in *Proc. IEEE VTS Asia–Pacific Wireless Commun. Symp. (APWCS)*, 2019, pp. 1–5.

[30] N. Qi, Z. Huang, W. Sun, S. Jin, and X. Su, "Coalitional formation-based group-buying for UAV-enabled data collection: An auction game approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 12, pp. 7420–7437, Dec. 2023.

[31] D. Wang, J. Ren, Z. Wang, Y. Wang, and Y. Zhang, "PrivAim: A dual-privacy preserving and quality-aware incentive mechanism for federated learning," *IEEE Trans. Comput.*, vol. 72, no. 7, pp. 1913–1927, Jul. 2023.

[32] D. Hui, L. Zhuo, and C. Xin, "Quality-aware incentive mechanism design based on matching game for hierarchical federated learning," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–6.

[33] J. Kang, Z. Xiong, D. T. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[34] D. M. Rosewater, R. Baldick, and S. Santoso, "Risk-averse model predictive control design for battery energy storage systems," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2014–2022, May 2020.

[35] C. A. Hans, P. Sopasakis, J. Raisch, C. Reincke-Collon, and P. Patrinos, "Risk-averse model predictive operation control of Islanded microgrids," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 6, pp. 2136–2151, Nov. 2020.

[36] J. Lu, Z. Zhang, J. Wang, R. Li, and S. Wan, "A green Stackelberg-game incentive mechanism for multi-service exchange in mobile crowdsensing," *ACM Trans. Internet Technol. (TOIT)*, vol. 22, no. 2, pp. 1–29, 2021.

[37] G. J. Mailath and L. Samuelson, *Repeated Games and Reputations: Long-Run Relationships*. Oxford, U.K.: Oxford Univ. Press, 2006.

[38] C. Hasan, "Incentive mechanism design for federated learning: Hedonic game approach," 2021, *arXiv:2101.09673*.

[39] S. Chu et al., "Design of two-level incentive mechanisms for hierarchical federated learning," 2023, *arXiv:2304.04162*.

[40] M. Aigner, "A characterization of the bell numbers," *Discr. Math.*, vol. 205, nos. 1–3, pp. 207–210, 1999.

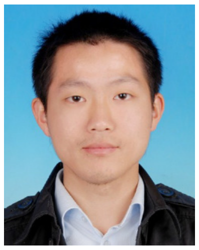
[41] Y. LeCun and C. Cortes. "The MNIST database of handwritten digits." 2005. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[42] G. Cohen, S. Afshar, J. C. Tapson, and A. van Schaik, "Emnist: Extending mnist to handwritten letters," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 2921–2926.

[43] M. Ayi and M. El-Sharkawy, "RMNv2: Reduced MobileNet V2 for CIFAR10," in *Proc. 10th Annu. Comput. Commun. Workshop Conf. (CCWC)*, 2020, pp. 287–292.

[44] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS*, 2011, pp. 1–9.

[45] W. Saad, Z. Han, M. Debbah, A. Hjørungnes, and T. Başar, "Coalitional games for distributed collaborative spectrum sensing in cognitive radio networks," in *Proc. IEEE INFOCOM*, 2009, pp. 2114–2122.
[46] M. Guazzone, C. Anglano, and M. Sereno, "A game-theoretic approach to coalition formation in green cloud federations," in *Proc. 14th IEEE/ACM Int. Symp. Clust., Cloud Grid Comput.*, 2014, pp. 618–625.



Gangqiang Hu received the B.S. degree from the Department of Computer Science and Engineering, Zhejiang Normal University, Jinhua, China, in 2011, where he is currently pursuing the Ph.D. degree with the College of Mathematics and Computer Science.

His research interests include federated learning, incentive mechanism, and game theory.

Jianmin Han received the B.S. and M.S. degrees from the Department of Computer Science and Technology, Daqing Petroleum Institute, Daqing, China, in 1992 and 2000, and the Ph.D. degree from the Department of Computer Science and Technology, East China University of Science and Technology, Shanghai, China, in 2009.

He is currently a Professor with the Department of Computer Science and Engineering, Zhejiang Normal University, Jinhua, China. His research interests include privacy preservation and game theory.

Jianfeng Lu received the Ph.D. degree in computer application technology from Huazhong University of Science and Technology, Wuhan, China, in 2010.

He was with Zhejiang Normal University, Jinhua, China, from 2010 to 2021, served as a Visiting Researcher with the University of Pittsburgh, Pittsburgh, PA, USA, in 2013, and is currently a Professor with the School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan. His research interests include federated learning, crowd computing, and game theory.

Juan Yu (Member, IEEE) received the M.S. degree in computer science from Zhejiang Normal University, Jinhua, China, in 2010, and the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2017.

She is currently an Associate Professor of Computer Science with the School of Computer Science and Technology, Zhejiang Normal University. Her research interests include spatio-temporal data mining, deep learning, and data privacy and security.

Sheng Qiu received the Ph.D. degree from East China Normal University, Shanghai, China, in 2023.

He is currently serving as a Lecturer with the School of Computer Science and Technology, Zhejiang Normal University, Jinhua, China. His research interests include deep learning and physically based simulation.

Hao Peng received the Ph.D. degree in computer science and technology from Shanghai Jiaotong University, Shanghai, China, in 2012.

He is currently a Full Professor with the School of Computer Science and Techonology, Zhejiang Normal University, Jinhua, China. He has published more than 100 papers in prestigious journal and conferences. His main research interests include AI security, IoT security, software and system security, data-driven security, and big data mining and analysis.

Prof. Peng served as a Program Committee Member for more than ten international conferences.

Donglin Zhu is currently pursuing the Doctoral degree in computer science and technology with Zhejiang Normal University, Jinhua, China.

He published more than 30 papers, and engaged in intelligent optimization algorithm, machine learning, image processing, and other related research.

Dr. Zhu is a reviewer for more than 40 international journals, such as *Expert Systems With Applications*, *Applied Soft Computing*, and *Swarm and Evolutionary Computation*.

Taiyong Li received the Ph.D. degree from Sichuan University, Chengdu, China, in 2009.

He is currently a Full Professor with the School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu. He has published over 80 papers in journals and conferences, including *Applied Soft Computing*, *Expert Systems with Applications*, NEUROCOM, *The Visual Computer Journal*, *Energy Conversion and Management*, *Multimedia Tools and Applications*, and CVPR. His research expertise lies in the fields of machine learning, computer vision, image processing, and evolutionary computation.