Auction-based Federated Learning using Software-defined Networking for resource efficiency

Eunil Seo
Department of Computing Science
Umeå University
Umeå, Sweden
eunil.seo@cs.umu.se

Dusit Niyato
School of Computer Science and Engineering
Nanyang Technological University
Singapore
dniyato@ntu.edu.sg

Erik Elmroth
Department of Computing Science
Umeå University
Umeå, Sweden
elmroth@cs.umu.se

***Abstract*—The training of global models using federated learning (FL) strategies is complicated by variations in local model quality arising from variation in data distribution across individual clients. A wide range of training strategies could be created by varying the size and distribution of the training data and the number of training iterations to be performed. All these variables affect both model quality and resource consumption. To facilitate the selection of good training strategies, we propose an auction-based FL method that can identify a training strategy that is optimal in terms of resource management efficiency subject to a given model quality requirement. An auction method is used to dynamically select resource-efficient FL clients and local models to minimize resource usage. This is enabled by using Software-defined Networking (SDN) to support the dynamic management of FL clients. We show that resource-optimal FL strategies can be implemented in the cloud/edge services market; dynamic quality-based model selection can reduce resource costs by up to 17% from the FL server's perspective. Moreover, the client utility function presented herein helps FL clients adopt practical trading strategies to cooperate efficiently with FL servers.**

***Index Terms*—Federated learning; cloud computing; software-defined networking; auction method; quality-based incentive**

I. Introduction

The advent of the age of data has made, Machine Learning, powerful computing resources, and large amounts of data available across the world's various cloud services, enabling the development of many diverse ML-based applications that exploit the vast numbers of interconnected cloud/edge data centers and mobile devices. In globally distributed data and resource environments, Federated Learning (FL) has emerged as a way of federating multiple ML models trained and distributed by many different model providers.

However, FL inevitably faces costs due to its reliance on distributed local ML models. Additionally, it suffers from model quality degradation when compared to centralized learning approaches. The auction-based FL strategy proposed here minimizes the use of computing resources by selecting appropriate data sizes and distributions. An overlay network is constructed from the chosen FL clients and an FL server; the server can then train a global model using the dynamic interactions supported by Software-Defined Networking (SDN).

Auction-based methods can help minimize resource costs by selecting resource-efficient FL clients from the perspective of the FL server. Such methods have been used in various applications including a radio network [1], a wireless FL service market [2], and a cellular wireless network [3]. The FL server acts as the auction buyer while the clients function as sellers. In this work, we extend the auction approach by introducing a winning score rule for the auctions that enables the selection of an optimal data size over multiple clients and a reward decision rule for selecting the highest quality local models during dynamic auction interactions.

To improve resource management, the FL server uses an SDN controller to create an overlay network in which the FL server and clients can perform auction bidding and product provision. This allows the FL server to adaptively control the resources to implement applications and services while satisfying quality requirements.

If the FL server has full knowledge of the quality of the available models and their training costs, it can select the best training strategies to minimize resources. To enable this, we define a two-tier model quality function that evaluates model quality based on the number of training iterations, data size, and distribution. With this function and the specified inputs, the FL server can predict model quality. In general, a greater number of iterations corresponds to a higher model quality but also increases the consumption of resources. Thus, the approach presented in this work allows resource usage to be minimized while satisfying an arbitrary quality requirement.

In this approach, the FL server uses the FedAvg algorithm [4] to integrate the selected local models. Auction-based FL allows both the FL server and the FL clients to maximize their utility. The main contributions of this work are:

- An auction method to reduce resource management in which a bidding process determines the optimal number of training iterations, data size, and data distribution.
- A two-tier model quality function that predicts computation resource use subject to ML model quality criteria and allows the FL server to minimize computation resource use.
- Utility metrics and utility functions to maximize the utility of the FL sever, FL client, and cloud provider. We also extend federated learning by using SDN and show that using an overlay network created by an SDN controller can improve resource management.

978-3-903176-36-2 ©2021 IFIP

The remainder of this paper is organized as follows. Section II presents related work. Section III describes the system models of the proposed auction-based FL. Section IV introduces the utility functions. Section V presents a performance evaluation. Finally, conclusions are presented in Section VI.

II. Related work

Due to the unique features of the local model in FL, previous studies have focused on two critical data attributes: the data size and data distribution. According to [5], data size plays an essential role in improving data quality; in general, more data means better predictive performance. With regards to the data distribution, Zhao *et al.* [6] claim that losses of accuracy are mainly due to divergence of weights. In this work, resource use is regarded as a model quality determinant of equal importance to the data size and distribution. This motivated the development of a two-tier model quality function based on these three factors: the data size, data distribution, and resource usage (i.e., the number of training iterations).

Another important determinant of the feasibility of FL is the reward method. Several reward methods have been developed: Niyato *et al.* [7] introduced the willingness-to-pay (WTP) approach, Han *et al.* [8] proposed a fairness-aware reward scheme, and Kang *et al.* [9] presented a reward method based on contract theory and the computational power of the available clients. In this work, the reward is proportional to the quality of the data used for model training; in other words, it depends on model quality and data size.

III. System model

This paper proposes a method for optimizing computational resource usage during federated learning while satisfying an arbitrarily chosen target accuracy requirement.

A. System model and auction procedure

The proposed federated learning system consists of an FL server (henceforth, the server), N FL clients (henceforth, clients) whose cloud/MEC computational resources are used to train local models, and an SDN controller (henceforth, the controller), as shown in Fig. 1. We denote the sets of clients and computation resources by $\mathbb{N}$ and $\mathbb{U}$, respectively. The corresponding numbers of clients and computation resources are N (indexed by n) and U (indexed by u).

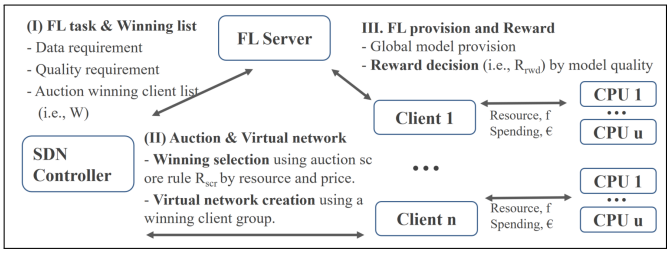

Fig. 1: System model for an auction-based federated learning system comprising an SDN controller, a server, clients, and cloud/edge providers.

The controller performs the auction and selects the winning client using the bidding resource and the claimed price. The server performs FL model training using the winning clients and judges the acceptability of local models based on their accuracy; only acceptable models are included in the global model and rewarded. The clients participate in the auction and train the local models if they win the auction; they are paid if they achieve acceptable accuracy. The auction-based FL process is divided into three stages, as shown in Fig. 1:

In stage I, the server designs the global model architecture and determines the data requirements (e.g., data type and sample size) and the required quality level (e.g., 97%). The server then requests a controller to conduct an auction using these requirements.

In stage II, a controller selects the winning clients using $R_{win}(\cdot)$ and creates an overlay network to improve resource management. We assume that clients are honest, i.e. that they use their local data and give the controller accurate information on their data size, computation resources, and claimed reward. The auction process is divided into the following sub-tasks:

(i) A controller defines and broadcasts **the auction winning rule** $R_{win}(\cdot)$, **the reward decision rule** $R_{rwd}(\cdot)$, and the FL training requirement to the distributed clients. The auction winning rule $R_{win}(\cdot)$ in (1) is used to decide the winning clients, the previously reported reward decision rule $R_{rwd}(\cdot)$ (7) is used to determine whether payment is awarded to the local model based on its quality, and the FL task requirements are used by clients when training local models.

(ii) A client n submits a bid $\mathbf{b}_n = \{l_n, \mathbf{e}_n, p_n, q_n\}$ to the controller. Here, l_n denotes the index of the global iterations, $\mathbf{e}_n = \{e_{n1}, e_{n2}, \ldots\}$ denotes the quality vector for the bidding resources, p_n denotes the claimed reward, and q_n denotes the local model quality after the l^{th} iteration.

(iii) After receiving all bids from the clients, a controller selects the set $\mathbb{W} = \{w(b_n)\}_{b_n \in \mathbb{B}}$ of all winning clients using the auction winning rule $R_{win}(\cdot)$ and determines the minimum number L_{min} of global iterations needed to satisfy the quality requirement Q_g using previously reported two-tier model quality functions (20).

(iv) The winner-based overlay network is created as shown in Fig. 2. Section III-B describes $R_{win}(\cdot)$ further.

In stage III, the clients train the local models and evaluate their accuracy; the server then uses the reward decision rule $R_{rwd}(\cdot)$ to determine payment and integrates the accepted local models. Federated learning thus involves a two-tier modeling structure consisting of local model training and model integration.

To determine the reward score $R_{rwd}(\cdot)$, the server averages the accuracies of all winning local models and subtracts the standard deviation according to the reward decision rule (7). The server uses the resulting reward score as a threshold value to select the best quality models and choose the accepted model group, $\{w_{l+1}^1, w_{l+1}^2, \ldots, w_{l+1}^k\}$.

This process in its entirety (i.e., going from stage I to stage III) constitutes one global iteration. The server iterates the global modeling process L_{min} (indexed by l) times, where the value of L_{min} is obtained using the two-tier model quality function (20). The notation used in the following discussion is explained in Table I.

TABLE I: Notation.

Symbol	Definition
$\mathbb{N}, N, n$	set of clients, number of clients, and client index
$\mathbb{U}, U, u$	set, number, and index of computation resources
Q_l	global model quality function, l is the global iteration index
Q_g	target global model quality requested by a server
Q_m	local model quality function, m is the local iteration index
w_l	a deep network at the l^{th} global iteration
$w_{l,m}$	a deep network at the m^{th} local & l^{th} global iteration
L, l	the global iteration number and the global iteration index
M, m	num. of local iteration for a given task, local iteration index
$R_{win}(\cdot)$	an auction winning rule
$R_{rwd}(\cdot)$	a reward decision rule
$\mathbf{b}$	a bid consisting of l, $\mathbf{e}$, claimed price p, model quality q
$\mathbf{e}$	a vector of the bidding resources: data, computation, etc.
$\mathbb{B}, \mathbb{W}, \mathbb{A}$	sets of **B**ids, **W**inning clients, **A**cceptable clients
$\mathbb{D}, D$	total dataset and dataset size of an FL server
$\mathbb{D}_n, D_n$	local dataset and data size of a client n

B. Auction winning score rule and overlay network creation

The auction winning score rule $R_{win}(\mathbf{e}, p)$ is a function of the bidding resources vector $\mathbf{e} = e_1, \ldots, e_i$ (e.g., data, CPU, etc.) and the claimed price p of the provisioning model per global iteration. As in previous work [10], the scoring function can be expressed as a quasi-linear function with coefficients $(\alpha_1, \ldots, \alpha_i)$, where i is the index of the different quality vectors. The winning scoring rule $r_{win}(\cdot)$ is used to compute the winning score as follows:

$$\begin{aligned} R_{win}(e_{n1}, \ldots, e_{ni}, p_n) &= r_{win}(e_{n1}, \ldots, e_{ni}) - p_n, \\ &= \alpha_1 e_{n1} + \ldots + \alpha_i e_{ni} - p_n, \end{aligned} \quad (1)$$

where n denotes the index of a client.

We present an example of an auction decision involving four clients $\mathbb{N} = \{A, B, C, D\}$, as shown in Fig. 2, and two resource types: the data size and frequency. The data size is assumed to vary in the range $[10, 100]$ while the frequency is the optimal frequency f^* as in (16), and is provided by the cloud provider.

The public winning scoring function is defined as $R(\mathbf{e}, p) = \alpha_1 e_1 + \alpha_2 e_2 - p$, where α_1 and α_2 are coefficients for the data size and the amount of computational resources. The weights assigned to the coefficients are 0.7 for α_1 and 0.3 for α_2.

In the initial stage of the global training, the four clients submit bids $b_n = (l, e_1, e_2, p, q)$: $b_A = (0, 13, 40, 21, \cdot)$, $b_B = (0, 17, 30, 20, \cdot)$, $b_C = (0, 14, 30, 21, \cdot)$, and $b_D = (0, 15, 30, 19, \cdot)$. After collecting them, the controller computes the following scores for each client: $R_{win}(b_A) = 21.1$, $R_{win}(b_B) = 20.9$, $R_{win}(b_C) = 21.9$, and $R_{win}(b_D) = 19.5$.

The controller notices that all of the claimed prices satisfy the winning score requirement and therefore sorts the four clients in descending order by mapping the winning scores and data sizes (e.g., $A : \{R_{win}(b_A) = 21.1, D_A = 13\}$, $B : \{20.9, 17\}$, $D : \{19.5, 14\}$, $C : \{18.8, 14\}$. Finally, the winning clients are selected based on the data size requirement. For instance, if the controller requires a data size of at least 40, $A : 13$, $B : 17$, and $D : 15$ are the winners, giving a total data size of 45, as shown in Fig. 2.

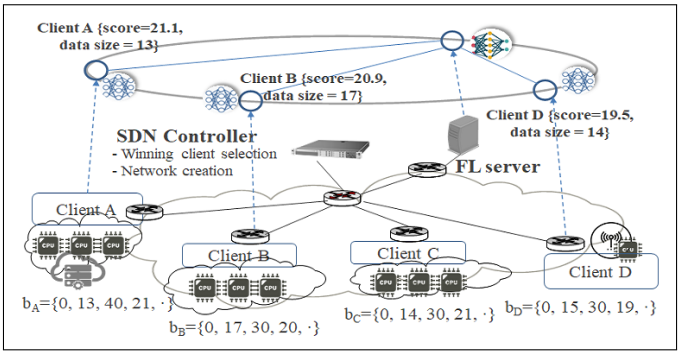

Fig. 2: The controller selects the winning clients using $R_{win}(\mathbf{e}, p)$ and creates an overlay network.

C. Two-tier model quality function

A local iteration begins when the winning clients receive the initial model $w_{l,m}$. Client n trains the initial model $w_{l,m}$ on its local dataset $\mathbb{D}_n$. After training, it returns the updated local model to the server. Previous studies [11] defined data utility functions in terms of the data size D_n as follows:

$$Q = \theta(D_n; q = [\alpha_1, \alpha_2]) = 1 - \frac{\alpha_1}{1 + \alpha_2 \cdot D_n}, \quad (2)$$

where α_1 and α_2 are curve fitting parameters.

Since this work focuses on optimizing computational resource use subject to a quality requirement, we extend these earlier model quality functions by introducing three variables: the maximum achievable accuracy (which depends on the data size), the training iteration, and the data distribution.

The maximum achievable accuracy is denoted by δ_m in (3) for the local model quality and δ_l in (4) for the global model quality. As the data size increases, the model accuracy improves; however, the model cannot achieve 100% accuracy even with unlimited data.

We define the local model quality function Q_m, which depends on the local iteration number m and a maximum achievable accuracy-related coefficient δ_m that in turn depends on the data size:

$$Q_m = \theta(m; p = [\delta_m, \alpha_1, \alpha_2]) = \delta_m \cdot \left(1 - \frac{\alpha_1}{1 + \alpha_2 \cdot m}\right), \quad (3)$$

where θ is a non-decreasing function with decreasing marginal accuracy.

A global iteration begins when the FL server receives all local models from the participating clients and integrates the gradients of the local models. Training of the global model continues until the FL server satisfies the quality requirement $L = \{1, \ldots, l\}$ times.

In keeping with Definition 4 of Oh et al. [12], we define the global model accuracy using the global iteration instead of the data size. As l increases (i.e., as the amount of computational resources grows), the global model accuracy Q_l improves within the limit of the maximum achievable accuracy δ_l:

$$Q_l = \psi(l; q = [\delta_l, \beta_1, \beta_2]) = \delta_l \cdot \left(1 - \beta_1 e^{-\beta_2 \cdot l}\right). \quad (4)$$

Unlike δ_m, δ_l is affected by both the data size D and the data distribution γ_d. The wider the distribution of the local models (i.e., the lower the γ), the lower the global model accuracy Q due to its distributed nature (See Fig. 3). We define δ_l as:

$$\delta_l = \Delta(D, \gamma), \quad (5)$$

where Δ is a coefficient that depends on the total data size D (e.g., $[0, 100]$) and the data distribution γ (e.g., $[0.01, 1]$).

Table II shows global model quality using different total data sizes, $\{10, \ldots, 100\}$ with identical data distributions ($\gamma = 0.1$, meaning that each client has 10% of the total data). The greater the data size, the higher the global model accuracy.

TABLE II: Model quality using different data sizes.

△(D, γ=0.1)	D=10	D=20	D=30	D=40	D=50	D=60	D=70	D=80	D=90	D=100
δ_l (%)	93.22	95.06	95.96	96.78	97.1	97.34	97.59	97.71	97.72	97.93

D. Quality-based rewarding and the reward decision rule

We propose a quality-based reward method using the model quality function. A higher local model accuracy may warrant a greater reward due to higher consumption of local resources. In accordance with Definition 5 of Oh et al. [12], we define the quality-based reward function by mapping the model accuracy to the local model payment:

$$p(b_n^l, m) = \begin{cases} \varsigma e^{-(1-Q_m(m))} & \text{if } a(b_n^l) = 1 \\ 0 & \text{otherwise,} \end{cases} \quad (6)$$

Here, $a(b_n^l) = \{1, 0\}$ is a binary variable indicating whether the model is acceptable or not, ς is a coefficient proportional to the contributed data size, m is the local iteration number, and Q_m is a local model quality function (3).

The server uses the local model accuracy to determine the payment for the local model: if the provided model quality exceeds the reward decision score $R_{rwd}(\cdot)$, a payment is made. We define the reward decision score as the average accuracy of the gathered local models minus the standard deviation:

$$R_{rwd}(q_1, \ldots, q_n, \beta_s, \sigma_s) = \frac{1}{n} \sum_{i=1}^{n} q_i - \beta_s \cdot \sigma_s. \quad (7)$$

Here, q_n is the accuracy of the local model provided by a client n, β_s is a coefficient to adjust the local model unacceptability scale, and σ_s is the standard deviation. The indicator variable $a(b_n^l)$ is determined as follows:

$$a(b_n^l) = \begin{cases} 1 & \text{if } q_n^l \geq R_{rwd}^l(\cdot) \\ 0 & \text{if } q_n^l < R_{rwd}^l(\cdot), \end{cases} \quad (8)$$

where q_n^l denotes the accuracy of the local model provided by client n and $R_{rwd}^l(\cdot)$ denotes the reward decision score.

E. Computation resource cost model

A client n receives support from a cloud provider in the form of a CPU set (which could, for example, be a set of frequencies $\{f_1, \ldots, f_u\}$). The client asks the cloud provider to train a local model by sending its local dataset $\mathbb{D}_n$ and Spending S_n. The cloud provider then allocates computation resources (the amount of which is determined by the value of S_n) to update the model $w_{l,m}$.

The cloud provider trains the local dataset $\mathbb{D}_n$ with a completion deadline τ_n. The task of the cloud provider is described using a commonly-used three-field task form (D_n, τ_n, X). The fields are the input-data size of the task D_n (in bits), the completion deadline τ_n (in seconds), and the computation workload/intensity X (in CPU cycles per bit).

To provide flexibility to accommodate diverse applications, we designed a computation cost model that estimates costs from an MEC server's perspective. For a client n, $f_{n,U_m} = \sum_{u=1}^{U} f_{n,u,m}$ denotes the total frequency of the allocated CPU units $\{1, \ldots, u\}$ at the m^{th} local iteration. We can thus obtain the total allocated frequency f_n over the full set of local iterations M by summing f_U:

$$f_n = \sum_{m=1}^{M} \sum_{u=1}^{U} f_{n,u,m} = \sum_{m=1}^{M} f_{n,U_m} = M \cdot f_{n,U}, \quad (9)$$

where $0 < f_{n,U} < f_n$ and M denotes the total number of the local iterations. For simplicity, we assume all local iterations have the same total frequency, i.e. that $f_{n,U_1} = \ldots = f_{n,U_m}$.

A cloud provider performs model training using computation resources $f_{n,U}$ during each local iteration. We can thus define the computation resource cost based on energy consumption as follows:

$$C_{n,U} = \mu \cdot E_{n,U}, \quad (10)$$

where μ is a predefined weight parameter for energy consumption and $E_{n,U}$ is the energy consumption for the local model.

In accordance with Equation 4 of Mao et al. [13], the energy consumption can be obtained from hardware data (e.g., the CPU frequency $f_{n,U}$) and the data size as follows:

$$E_{n,U} = k \cdot D_n \cdot X \cdot f_{n,U}^2, \quad (11)$$

where k is a constant related to the hardware architecture.

An FL client n calculates the computation cost associated with a given number of local iterations M as follows:

$$C_n = \sum_{m=1}^{M} C_{n,U} = M \cdot \mu \cdot k \cdot D_n \cdot X \cdot f_{n,U}^2, \quad (12)$$

which allows us to define the total computation cost of M local iterations for an FL client n.

IV. Non-cooperative competition and utility optimization

We formulate a non-cooperative competition game between a client and a cloud provider in which both stakeholders perform their roles while seeking to maximize their utility. Furthermore, we propose utility functions to maximize the profitability of the client and server.

A. The computation resource utility

During a given global iteration l, a client n plans a given amount of Spending $S_n = \sum_{m=1}^{M} S_{n,U_m}$ to purchase computation resources from a cloud provider. The cloud provider computes its spending during each local iteration m as:

$$S_{n,U_m} = \frac{f_{n,U_m}}{\sum_{m=1}^{M} f_{n,U_m}} \cdot S_n. \tag{13}$$

Here, f_{n,U_m} denotes the allocated CPU frequency during a local iteration m, $\sum_{m=1}^{M} f_{n,U_m}$ denotes the sum of all CPU frequencies during a global iteration l, and S_n denotes the spending of client n during a global iteration.

We can define **the computation resource utility** using the relationship between the revenue (i.e., S_{n,U_m}) and the computation resource cost (i.e., f_{n,U_m}):

$$U_u(f_{n,U_m}, \gamma_m) = S_{n,U_m} - \gamma_m \cdot f_{n,U_m}. \tag{14}$$

In keeping with a previously proposed approach [14], if all CPUs participate in each local iteration, each CPU u is allocated and used (i.e., $f_u > 0, \forall u$). We impose the conditions that $U > 1$, $M > 1$, $\sum_{j \neq u}^{U} \gamma_j > (U-2)\gamma_u$, where U denotes the number of participating CPU units and all γ_u are assumed to be equal. We can then define the optimal CPU frequency for each local iteration:

$$f_u^* = \frac{S_n(U-1)(\sum_{i=1}^{U} \gamma_i - (U-1)\gamma_i)}{(\sum_{i=1}^{U} \gamma_i)^2}, \forall u. \tag{15}$$

Finally, we can sum the CPU frequencies used by the cloud provider during a single local iteration and simplify the expression to obtain:

$$f_U^* = \sum_{i=1}^{U} f_i^* = \frac{S_n(U-1)}{\sum_{i=1}^{U} \gamma_i}. \tag{16}$$

B. The FL client utility

Each client aims to maximize its own utility and will not join the federated learning system if the reward is insufficient compared to the costs of data and computation resources. This work focuses on optimizing computation resource usage; therefore, **the client utility** is defined as the reward price minus data and spending costs:

$$U_n(b_n^l) = \begin{cases} p(b_n^l) - C_{D_n} - S_n & \text{if a bid } b_n^l \text{ wins} \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

where $p(b_n^l)$ denotes a reward with a bid (i.e., b_n^l) in (6), C_{D_n} denotes a local data cost, and S_n denotes the total spending on computation resources in (12) during the l^{th} global iteration.

$$U_n = a(b_n^l) \cdot \varsigma e^{-(1-Q_m(m))} - \gamma_n D_n - M\mu k D_n X f_U^{*2}, \tag{18}$$

where γ_n is the data generation coefficient, M is the number of local iterations, and f_U^* is the optimal frequency in a given local iteration from the cloud provider's perspective.

C. The FL server utility

We define **the server utility** as the model quality minus the total rewards granted to the winning clients:

$$\begin{aligned} U_s &= Q_l - \sum_{b_n^l \in \mathbb{A}} p(b_n^l), \\ &= \delta_l \cdot \left(1 - \beta_1 e^{-\beta_2 \cdot l}\right) - \sum_{b_n^l \in \mathbb{A}} a(b_n^l) \cdot \varsigma e^{-(1-Q_m(m))}, \end{aligned} \tag{19}$$

where Q_l represents the global model quality defined in (4), $p(b_n^l)$ denotes the reward price for a bid b_n^l as defined in (6), and $\mathbb{A} = \sum_{i=1}^{L} A_i$ is the set of acceptable clients.

Once the target accuracy Q_g is known, computation resource usage can be minimized by selecting the minimum number of global iterations. For instance, if the target accuracy is $Q_g = 95\%$, the server trains the global model for 61 iterations only (See $D = 30$ and $\gamma = 0.1$ in Fig. 3).

The FL server can determine the minimum number of iterations as the global model quality minus the required accuracy Q_g:

$$\begin{aligned} \arg\min_l \delta_l \cdot \left(1 - \beta_1 e^{-\beta_2 \cdot l}\right) - Q_g \geq 0, \delta_l \geq Q_g, \\ \text{s.t.} \quad l \geq \frac{log\frac{\delta_l \cdot \beta_1}{\delta_l - Q_g}}{\beta_2}, \end{aligned} \tag{20}$$

where $\delta_l = \Delta(D, \gamma)$ is the maximum achievable accuracy of the global model for the given D and γ according to (5).

When training the global model, the server makes a set of choices $\mathbb{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$, where $\mathbf{c}_1 = \{l_1, m_1\}$ is a pair of global and local iteration numbers. The server maximizes its utility by minimizing the cost of obtaining the target model quality Q_g:

$$U_s^{Q_g} = \min \sum_{l=1}^{L} \left(\min \sum_{b_n^l \in \mathbb{A}_l} a(b_n^l) \cdot \varsigma e^{-(1-Q_m(m))} \right), \tag{21}$$

where $Q_m(m)$ denotes the local model accuracy achieved after m local iterations. By comparing the costs of several choices, $\mathbb{H}$, the server can maximize its profitability.

V. Performance Evaluation

We verify the model quality function described above and demonstrate that our proposed federated learning strategy enables optimal resource management by performing simulations with the classical MNIST dataset, which contains 50,000 training samples and 10,000 testing samples representing handwritten digits from 0 to 9.

A. Verification for the model quality function

In this work, the data size, data distribution, and number of training iterations determine the global model quality and serve as the inputs for the global model quality function defined in (4). To verify the model quality function, we train multiple global models with the following simulation parameters: the learning rate μ is set to 0.01, the number of global epochs is $l = \{1, \ldots, 400\}$, and the number of local epochs is $m = \{1, 2, 5, 10\}$.

To evaluate how the data size affects model accuracy, we use the FedAvg algorithm; Table III shows the maximum achievable accuracy for data sizes between 10 and 100. It is clear that the greater the data size, the higher the model accuracy. However, the improvement in accuracy may become saturated once the data size exceeds a certain threshold.

TABLE III: The maximum achievable accuracy.

△(D, γ)	10, 0.1	20, 0.1	30, 0.1	40, 0.1	50, 0.1	60, 0.1	70, 0.1	80, 0.1	90, 0.1	100, 0.1
δ_l (%)	93.22	95.06	95.96	96.78	97.1	97.34	97.59	97.71	97.72	97.93
β1	0.0248	0.026	0.0423	0.044	0.0444	0.0519	0.0468	0.0467	0.061	0.0654
β2	0.0903	0.0979	0.1156	0.0957	0.0867	0.0891	0.0783	0.0735	0.0794	0.0753

The maximum achievable accuracy $\delta_l = \Delta(D, \gamma)$ and the values of the curve-fitting parameters β_1 and β_2 for global models with 10 different data sizes.

A wide data distribution means that the total data set consists of a large number of relatively small split data sets. For instance, if $\gamma = 0.1$ then each client uses only 10% of the total data size. Lower values of γ corresponding to wider data distributions are associated with lower accuracy, as shown in Fig. 3. For example, in the case of a model with a data size of $D = 20$, the maximum achieved accuracy is 95.39% when $\gamma = 0.5$ but 94% when $\gamma = 0.025$.

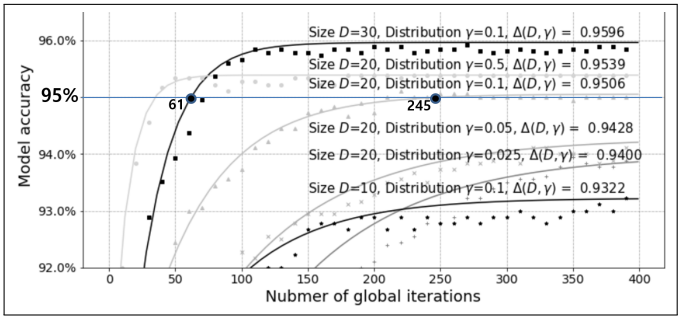

Fig. 3: Model accuracy for a data size of $D = \{10, 20, 30\}$ with different data distributions, $\gamma = \{0.5, 0.1, 0.05, 0.025\}$.

B. Server utility maximization

The server's utility is improved by reducing computation resource usage. This requires determination of the minimum number of global training iterations because each iteration increases resource usage. The server can use the global model quality function Q_l in (4) to determine the required data size and the minimum number of global training iterations needed to satisfy the quality requirement Q_g, as shown in Fig. 3.

For instance, if a quality requirement of 95% is specified, a model with the distribution $\Delta(30, 0.1)$ can achieve it after 61 iterations (i.e., $L_{min} = 61$), whereas an alternative model with $\Delta(20, 0.1)$ has an L_{min} of 245 for the same quality requirement.

The server computes the total reward price using L_{min} as defined in (20) and $U_s^{Q_g}$ as defined in (21). Table IV shows the total reward price and the corresponding L_{min} values for global models trained using ten different data sizes $\{10, \ldots, 100\}$ with a data distribution parameter of $\gamma = 0.1$. This shows that the greater the data size, the higher the model accuracy and the more quickly the accuracy increases.

TABLE IV: Total reward price and L_{min} of global models.

Target:96%	10, 0.1	20, 0.1	30, 0.1	40, 0.1	50, 0.1	60, 0.1	70, 0.1	80, 0.1	90, 0.1	100, 0.1
Accuracy,%	N/A	N/A	N/A	96.02	96.01	96.01	96.03	96.03	96.01	96.03
Iteration	∞	∞	∞	57	46	36	34	30	25	21
Reward	∞	∞	∞	2176	2203	2061	2283	2301	2158	2018

Fig. 4 shows the reward prices for different model quality requirements. Using the equations presented above, the server can estimate the reward price for a given quality requirement and then determine the utility maximizing optimal data size, data distribution, and number of global iterations.

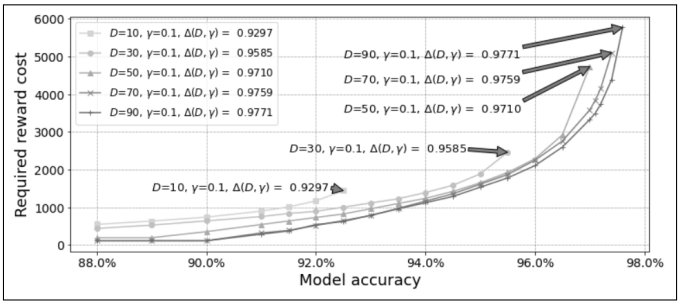

Fig. 4: Reward price as a function of model accuracy.

Fig. 5 shows the server utility, which is defined as the achieved quality requirement minus the total reward price. The greater the data size, the better the server utility. It is worth noting however that above a certain threshold, increasing the data size reduces the server's utility because increasing model accuracy also increases the quantity of computational resources needed for model training.

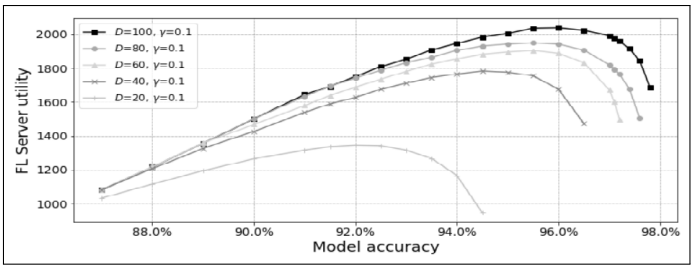

Fig. 5: The server utility based on all participating clients.

Fig. 6 compares the accuracy of a model trained with the full-size dataset to that achieved after using the reward score function defined in (7) to remove unacceptable models (e.g., lower quality) in order to minimize reward prices. Lower values of the local model unacceptability parameter β_s result in higher rejection rates and thus greater cost savings.

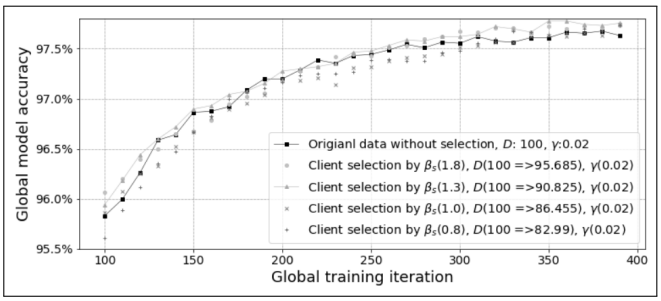

Fig. 6: Accuracy using the original dataset and after applying the reward score function.

The results obtained show that the server can reduce reward prices by up to 17% while maintaining model accuracy; the root mean square error (RMSE) obtained after applying the reward score function is 0.007, as shown in Table V.

TABLE V: RMSE and R-squared values.

β_s	Full size	1.8	1.3	1	0.8
Data size	100	95.68	90.82	86.46	82.99
RMSE	0	0.0051	0.0078	0.0041	0.0051
R-squared	1	0.9906	0.9754	0.9949	0.9908

C. Client utility maximization

A client maximizes its utility U_n by maximizing its reward while minimizing its data and computation resource costs according to (18). Fig. 7 shows the client utility, which is the reward minus the data and computation resource cost. The global model accuracy indicates that client utility is maximized when a client performs only two training iterations with its local dataset. While ten local iterations give the highest accuracy, it also incurs a greater computation resource cost, leading to an overall reduction in client utility.

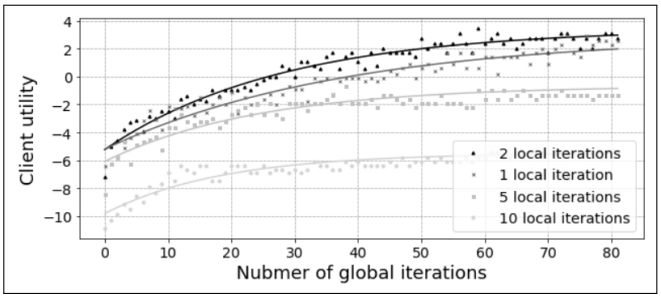

Fig. 7: Client utility after 1, 2, 5, and 10 local iterations.

Table VI shows the utility of each client when performing two local iterations. The rewards of each client differ because the local models vary in quality despite having identical data and computation resource costs. Each client can plan its model training based on computation resource costs.

TABLE VI: Utility of ten clients after two local iterations.

Client ID	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Reward	645.2	720.09	742.12	659.76	665.02	702.16	626.01	698.81	606.88	698.36
Data cost	450	450	450	450	450	450	450	450	450	450
Spending	147.6	147.6	147.6	147.6	147.6	147.6	147.6	147.6	147.6	147.6
Utility	47.6	122.49	144.52	62.16	67.42	104.56	28.41	101.21	9.28	100.76

VI. Conclusion

We have extended federated learning by combining the auction method and SDN management to create a practical trading strategy for an FL server and clients in the cloud/edge service market. This work demonstrates that the FL server maximizes its utility by minimizing the number of global training iterations and minimizing computation costs by computing a reward decision score. The client maximizes its utility by performing an optimal number of training iterations determined by comparing the reward to its resource cost. We will extend this work by using the proposed overlay network to achieve reliable and scalable resource-efficient Service Function Chaining (SFC).

Acknowledgment

This work was partly supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] X. Gao, P. Wang, D. Niyato, and et al, "Auction-based time scheduling for backscatter-aided rf-powered cognitive radio networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, 2019.

[2] Y. Jiao, P. Wang, D. Niyato, B. Lin, and D. I. Kim, "Toward an automated auction framework for wireless federated learning services market," 2020.

[3] T. H. T. Le, N. H. Tran, and et al, "Auction based incentive design for efficient federated learning in cellular wireless networks," in *2020 IEEE Wireless Communications and Networking Conference*, 2020.

[4] H. B. McMahan, E. Moore, D. Ramage, and et al., "Communication-efficient learning of deep networks from decentralized data," 2017.

[5] P. Domingos, "A few useful things to know about machine learning," p. 78–87, oct 2012.

[6] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018.

[7] D. Niyato, M. A. Alsheikh, P. Wang, and et al., "Market model and optimal pricing scheme of big data and internet of things (iot)," 2016.

[8] H. Yu, Z. Liu, Y. Liu, T. Chen, M. Cong, X. Weng, D. Niyato, and Q. Yang, "A sustainable incentive scheme for federated learning," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 58–69, 2020.

[9] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim, "Incentive design for efficient federated learning in mobile networks: A contract theory approach," 2019.

[10] R. Zeng, S. Zhang, J. Wang, and X. Chu, "Fmore: An incentive scheme of multi-dimensional auction for federated learning in mec," 2020.

[11] W. E. Armstrong, "The determinateness of the utility function," pp. 453–467, 1939.

[12] H. Oh, S. Park, G. M. Lee, J. K. Choi, and S. Noh, "Competitive data trading model with privacy valuation for multiple stakeholders in iot data markets," *IEEE Internet of Things Journal*, vol. 7, no. 4, 2020.

[13] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[14] B. Jang, S. Park, and et al., "Three hierarchical levels of big-data market model over multiple data sources for internet of things," 2018.