Privacy vs. Efficiency: Achieving Both Through Adaptive Hierarchical Federated Learning

Yeting Guo, Fang Liu, Tongqing Zhou, Zhiping Cai, and Nong Xiao

***Abstract*—As a decentralized training paradigm, Federated learning (FL) promises data privacy by exchanging model parameters instead of raw local data. However, it is still impeded by the resource limitations of end devices and privacy risks from the 'curious' cloud. Yet, existing work predominately ignores that these two issues are non-orthogonal in nature. In this article, we propose a joint design (i.e., AHFL) that accommodates both the efficiency expectation and privacy protection of clients towards high inference accuracy. Based on a cloud-edge-end hierarchical FL framework, we carefully offload the training burden of devices to one proximate edge for enhanced efficiency and apply a two-level differential privacy mechanism for privacy protection. To resolve the conflicts of dynamical resource consumption and privacy risk accumulation, we formulate an optimization problem for choosing configurations under correlated learning parameters (e.g., iterations) and privacy control factors (e.g., noise intensity). An adaptive algorithmic solution is presented based on performance-oriented resource scheduling, budget-aware device selection, and adaptive local noise injection. Extensive evaluations are performed on three different data distribution cases of two real-world datasets, using both a networked prototype and large-scale simulations. Experimental results show that AHFL relieves the end's resource burden (w.r.t. computation time 8.58% ↓, communication time 59.35%↓ and memory consumption 43.61%↓) and has better accuracy (6.34%↑) than 3 typical baselines under the limited resource and privacy budgets. The code for our implementation is available at https://github.com/Guoyeting/AHFL.**

***Index Terms*—Distributed machine learning, edge computing, federated learning, privacy protection.**

Manuscript received 7 June 2022; revised 30 November 2022; accepted 7 February 2023. Date of publication 13 February 2023; date of current version 7 March 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFC2003404, in part by the National Natural Science Foundation of China under Grants 62172155, 61832020, 62072465, and 62102425, in part by the Natural Science Foundation of Guangdong Province under Grant 2018B030312002, in part by Science and Technology Innovation Program of Hunan Province under Grants 2021RC2071, and 2022RC3061, and in part by the National Natural Science Foundation of China-Guangdong Joint Fund under Grant NSFCU1811461. Recommended for acceptance by J. Wang. *(Corresponding authors: Fang Liu; Nong Xiao.)*

Yeting Guo, Tongqing Zhou, and Zhiping Cai are with the College of Computer, National University of Defense Technology, Hunan 410073, China (e-mail: guoyeting13@nudt.edu.cn; zhoutongqing@nudt.edu.cn; zpcai@nudt.edu.cn).

Fang Liu is with the School of Design, Hunan University, Hunan 410082, China (e-mail: fangl@hnu.edu.cn).

Nong Xiao is with the College of Computer, National University of Defense Technology, Hunan 410073, China, and also with the School of Computer Science and Engineering, Sun Yat-sen University, Guangdong 510000, China (e-mail: nongxiao@nudt.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPDS.2023.3244198, provided by the authors.

Digital Object Identifier 10.1109/TPDS.2023.3244198

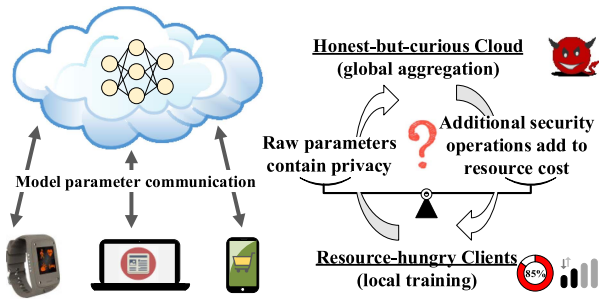


Fig. 1. Resource and privacy concerns in FL.

I. Introduction

THE evolving mobile computing technologies have generated tremendous data at the pervasive smart devices. Its marriage with advanced AI algorithms has nurtured a wide spectrum of applications in healthcare, mobile social networks, smart cities, etc. However, users' awareness of personal data privacy significantly hinders the centralized gathering and exploitation of such fruitful data. Recently, Federated Learning (FL) has ascended to the spotlight by providing a solution for distributed data owners to collaboratively learn a model in parallel, while keeping their sensitive data locally [1], [2]. In fact, FL has been applied in products for intelligent traffic and virtual assistants [3].

The modern FL architectures typically adopt a cloud-end framework. Wherein, local training and global aggregation are performed alternately and iteratively at the end and cloud sides, respectively, for model optimization. However, as shown in Fig. 1, a practical implementation of such a framework will unavoidably face the concerns of *resource* and *privacy* raised by the two parties.

Relatively 'Weak' End Devices. The training of complex models containing multiple multi-unit network layers can easily drain the on-board CPU and even impact the running performance of clients' mobile devices [4]. Meanwhile, the network environments, especially the uploading link, of end devices fluctuate dynamically, so frequent local model uploading is communication inefficient [5].

'Honest-but-Curious' Cloud. The cloud honestly aggregates the model updates of end devices, but out of curiosity, is still capable of recovering the raw training data, thus inferring the sensitive information from users [6].

As two major hindrances to FL, the above two issues are widely, while independently, studied in existing work. On one

hand, some have proposed to optimize the schedule of local training and global communication tasks to improve resource utilization [7], [8], [9]. Also, dedicated DNN compression techniques are proposed to reduce the communication burden during computation offloading [10]. On the other hand, to enhance privacy, homomorphic encryption, multi-party computation and differential protection (DP) mechanisms are widely applied [11]. Wherein, DP is more favored for lightweight computation [12], [13], while its privacy guarantee will change dynamically given various factors in FL, such as the communication rounds. We argue that efficiency and data privacy of FL are in fact non-orthogonal from the perspective of model training. Purely pursuing efficiency may enlarge the threat surface of local data, while the incurred overhead or noise of privacy protection would further degrade FL's efficiency and performance.

In this paper, we investigate the benefit of involving edge computing [14] to relieve the resource and threat concerns simultaneously. For this, we first propose a cloud-edge-end hierarchical FL framework. Under such a framework, the end is only responsible for data collection and partial model training task; the edge (e.g., a base station), for its physical proximity and relatively stronger computing capabilities, is involved to assist the end for local model training based on model partition [15], thus relieving the end's computation. Meanwhile, we utilize a two-level DP mechanism for privacy consideration in our hierarchical FL system. First, we perturb the model updates between the cloud and network edge (including the end device and the edge server), to resist the cloud and external attackers from identifying or even rebuilding the local training samples. Besides, we propose to perturb the intermediate features [16] from the end to the edge, which prevents privacy leakage during the computation offloading. In this way, a full chain of privacy protection can be constructed.

Yet, the integration of privacy protection into a hierarchical framework is not that straightforward, as it seems to be, in practice. The technical challenges are two-fold:

- To efficiently attain high inference accuracy, we need to handle the optimal configuration searching problem for a series of inter-correlated parameters. For the cloud, both the resource schedule for local and global iterations and the device sampling rate will impact the DP protection strength. For the end devices, offloading decision and noise intensity setting have to be made given resource overhead and privacy constraints. It's necessary to coordinate their internal connection carefully.
- The optimal configuration varies with the resource exploitation and the risk accumulation caused by overlapping participants, and is heterogeneous under different data distributions. As a result, a generic and empirical setting is not adequate. It's necessary to dynamically accommodate the constraints and optimize the model.

Given these challenges, we further formulate the model optimization problem under both the resource budget and privacy constraints, and propose an Adaptive control algorithm for the above Hierarchical FL framework, AHFL. First, by monitoring the resource consumption, as well as the distributed training state in real-time, we adjust the resource allocation of local training and global communication after each aggregation. In this way, resource utilization is tuned for a better gain of model performance improvement. Second, based on the scheduling, the number of remaining aggregation rounds and the corresponding privacy risk accumulation are estimated, which will guide the adjustment of the device sampling rate. By doing this, the algorithm finds the optimal critical point of both constraints. Third, based on the updated global configuration and local privacy constraint, an end device determines whether to offload the task to the edge. It strikes to minimize the local resource consumption by controlling the injected noise intensity.

In summary, our main contributions are listed as follows:

- We propose a cloud-edge-end FL system to satisfy the efficiency expectation and privacy constraints of FL participants in one shot. It exploits the hierarchical computation resources to reduce the burden of end devices, and applies a two-level DP protection mechanism to provide a full chain of privacy protection.
- We propose an adaptive control algorithm for the hierarchical system to minimize loss value under both the quantified resource requirements and privacy constraints. It learns system dynamics and training state in real-time, and then dynamically adjust resource scheduling and privacy configurations towards high inference accuracy.
- We implement a prototype for AHFL and conduct extensive evaluations based on two real-world datasets. The results demonstrate the performance superiority (w.r.t computation time 8.58% ↓, communication time 59.35%↓, memory consumption 43.61%↓ and inference accuracy 6.34%↑) of AHFL over other state-of-the-art approaches (i.e., ltAdap [8], FLGDP [17], FEEL [18]).

This paper extends our preliminary conference version FEEL [18] with the consideration of 'curious' cloud. Moreover, in this paper, we design a novel adaptive control mechanism for the hierarchical framework (see Section IV). It discusses the correlation between the resource efficiency optimization and privacy strengthening. And it achieves higher inference accuracy than FEEL by dynamic adjustment.

II. Related Work

In this section, we discuss the related work on resource efficiency and privacy leakage under FL framework.

Efficient FL. To reduce the local training burden, Niu et al. and Jia et al. designed federated submodel learning schemes which allow the end to train only the related parts of the full models [22], [23]. Edge computing [24], [25] proposes to leverage resources at the network edge to optimize computation task execution. Motivated by this paradigm, FEEL offloads partial FL computation task from the end to the edge server to alleviate local computation pressure [18]. To reduce the communication traffic, Jin et al. designed an online participant selection learning algorithm to exclude unnecessary model updates [26]. Wang et al. divided the devices into clusters and aggregated model parameter updates hierachically [7]. Some compression methods like clipping and deduplication are also applied to reduce the communication [27], [28]. These works empirically set a fixed global

TABLE I
COMPARISON WITH THE STATE-OF-THE-ARTS

Method		Mao *et al.*, 2017 [16]	Bonawitz *et al.*, 2019 [19], Line *et al.*, 2021 [20]	Wang *et al.*, 2019 [8], Chen *et al.*, 2021 [9]	Geyer *et al.*, 2017 [17], Huang *et al.*, 2020 [21]	Guo *et al.*, 2020 [18]	Our Scheme
Entity	End Device	Training*	Training	Training	Training	Training*	Training*
	Edge Server	Training	×	×	×	Training	Training*
	Cloud Center	×	Aggregation	Aggregation	Aggregation*	Aggregation*	Aggregation
Function	Resource Scheduling	Fixed	Fixed	Adaptive	Fixed	Fixed	Adaptive
	Privacy Protection	Fixed	×	×	Fixed	Fixed	Adaptive
Performance	Resource Consumption	Small	Large	Medium	Large	Medium	Small
	Attack Resistance	Edge Server	×	×	External Attacker	External Attacker and Edge Server	External Attacker, Edge and Cloud
	Inference Accuracy	Poor	Good	Good	Medium	Medium	Good

* It refers to the privacy protection apart from basic identity authentication and data encryption.

aggregation frequency, that is, how many training iterations between two adjacent aggregations. They ignore the task scheduling of local training and global aggregation, which may be hard to converge the model under a certain resource budget. Wang et al. dynamically controlled the relationship between the two FL tasks to minimize the loss value in real time [8]. Li et al. supported flexible communication compression and balances the energy consumption of FL tasks [9]. However, these works still neglect an important restriction (i.e., the privacy issue) while purely pursuing efficiency may enlarge the threat surface of local data.

Privacy Issues of FL. Various attacks on the learning model have been proposed [29]. Carlini et al. demonstrated individual training sample recovery by querying the language model GPT-2 [30]. Song et al. analyzed the user-level privacy attacks against FL [31]. Many researchers are committed to finding corresponding solutions. They are mainly categorized into encryption methods [11], [32], such as Homomorphic Encryption, and perturbation methods, such as Differential Privacy (DP). The former is always resource-consuming for mobile devices to perform complex encryption and decryption operations. The latter is relatively more applicable to network edge. DP has been used in each stage of model learning, including data collection, gradient computation, parameter uploading and result prediction [33]. But DP may damage the model inference performance. A trade-off between model utility and privacy protection is required. Mao et al. proposed a DP-enhanced offloading strategy to protect from the membership attack of the edge [16]. It discusses the offloading task selection, and makes a balance between resource consumption and model performance with privacy guarantees. Geyer et al. designed client sided DP preserving federated optimization to hide clients' contributions from the attacker [17]. Hu et al. considered the device heterogeneity and proposed a personalized FL scheme with DP [34].

Different from the above existing efforts, we accommodate both the efficiency expectation and privacy requirements towards high inference accuracy. Table I indicates a comprehensive comparison with the state-of-the-arts from the role of system entity, control function design and system performance. The main differences are summarized as follows: 1) we design a cloud-edge-end hierarchical FL system to improve efficiency, and apply DP protection as close to the end as possible to provide a full chain of privacy protection; 2) we argue that the efficiency and data privacy of FL are non-orthogonal in model optimization, and achieve a desirable trade-off among the three performance metrics in FL by adaptively coordinating the resource optimization and privacy protection operations.

III. PRELIMINARIES AND SYSTEM OVERVIEW

A. Preliminaries on FL

Generally, FL involves two entities, the cloud and the end. It is essentially a circular execution of the following steps.

Step 1. The cloud initializes a global model if there is no pre-saved model, the global model parameters in iteration t are denoted as $w(t)$. It randomly selects a subset of available devices to send $w(t)$.

Step 2. The device i initials the local parameters $w_i(t)$ via $w(t)$. Then it updates $w_i(t)$ with local dataset D_i by stochastic gradient descent algorithm. After a certain training iterations τ, the device i sends $w_i(t+\tau)$ to the cloud.

Step 3. The cloud aggregates these distributed $w_i(t+\tau)$ and updates global $w(t+\tau)$ by the FedAvg algorithm in [19].

In this paper, K is the number of global aggregation rounds, and $T = K \cdot \tau$ refers to the total number of iterations.

B. Concerns on Resources and Privacy

Resource Concerns. The end device independently trains the local model in FL. Due to the limited computing capability, it is difficult to train complex models efficiently. We applied the same model in a laptop and a raspberry pie respectively (details in Section V-A), and evaluated the local training time. We found that the time for the raspberry pie increased by $50\times$ than the laptop. Parameter communications are also time-consuming. For example, the wide-used Resnet-50 model contains approximately 100MB parameters [9]. It conflicts with limited communication

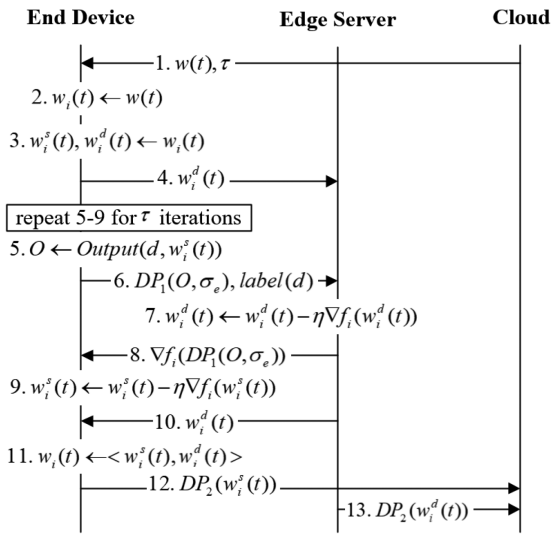


Fig. 2. Workflow of training task offloading.

resources at the end and causes bandwidth bottlenecks in core networks.

Threat Model. There exist various attacks towards FL. In this paper, we hold the security assumption that the cloud is honest-but-curious. It honestly aggregates $w_i(t)$, but also attempts to infer sensitive information from $w_i(t)$. The outside adversary can also perform the inference attack by intercepting their communications. For example, a certain commodity corresponds to a specific unit of the embedding layer. It's identifiable whether one user has purchased the commodity through the updates on the unit [35].

It's challenging to balance the two concerns. Existing resource optimization strategies often neglect the fact that raw parameters contain privacy, and some of them even enlarge the privacy leakage, resulting in unavailability, while additional security operations increase resource costs.

C. Our Hierarchical FL Design

In view of the two concerns, we propose a cloud-edge-end Hierarchical FL system (HFL). It optimizes the classical FL with the following designs.

1) Edge-Assisted Training Optimization: We extend the origin FL with the assistance of edge computing. The core idea is to offload computation tasks to some network nodes to alleviate the above resource concerns. These nodes are called edge servers, which are usually close to the end and have strong computing capabilities, such as base stations. The workflow is shown in Fig. 2.

The end device i receives $w(t)$ and τ from the cloud and initializes the local $w_i(t)$ to $w(t)$ (step 1-2). The device i splits the model into two parts, shallow layers and deep layers (step 3). Their corresponding model parameters are denoted as $w_i^s(t)$ and $w_i^d(t)$. The device sends $w_i^d(t)$ to the edge server in proximity, and the edge server initials the deep network layers with $w_i^d(t)$ (step 4). There are many model splitting strategies [18], [36]. They are compatible with our framework, and we will not discuss the splitting strategy in detail. Then in the next τ iterations, the end calculates the output of each batch d in the shallow network layers (step 5). The output O is also called intermediate features of the model. The device sends the perturbed features $DP_1(O, \delta_e)$ and the corresponding $label(d)$ to the edge in turn (step 6).

The edge regards the perturbed features as its input and learns the deep layers through gradient descent (step 7). And it also computes the gradient of loss function f_i to the perturbed features to assist the end to coordinate the shallow layers (step 8-9). After training, the edge sends $w_i^d(t)$ to the device, and the device updates $w_i(t)$ with the updated $w_i^s(t)$ and $w_i^d(t)$ (step 10-11). For global model aggregation, the device and the edge send noised $DP_2(w_i^s(t))$ and $DP_2(w_i^d(t))$ to the cloud, respectively (step 12-13). The function DP_1 and DP_2 would be introduced later.

2) DP-Strengthen Privacy Preservation: For privacy enhancement, we apply a two-level DP protection mechanism. It could maximize the data analytical availability and minimize the chance of individual privacy leakage. Here we introduce some related basics first.

DP: A mechanism M with domain $\mathbb{D}$ and range $\mathbb{R}$ satisfies (ϵ, δ)-DP for two non-negative numbers ϵ and δ if for any pair of adjacent datasets $< d, d' >$, for any subset of outputs $s \in \mathbb{R}$, it holds that

$$Pr\{M(d) \in s\} \leq e^{\epsilon} Pr\{M(d') \in s\} + \delta \tag{1}$$

Adjacent datasets are two subsets of $\mathbb{D}$ whose difference is reflected in only one single record. ϵ indicates the privacy loss of DP, also called privacy budget, and δ indicates the probability that original ϵ-DP is broken. The smaller the value of ϵ, the greater the probability of obtaining the same result for adjacent datasets, and the smaller the risk probability that the attacker can identify whether the single target record is in the dataset. We use ϵ to quantify the privacy constraint in our subsequent adaptive control design.

Gaussian Mechanism. To achieve (ϵ, δ)-DP, it builds a new noised function M, which is the sum of the original query function f_{qr} and a random noise obeying normalized distribution $N(0, S_f^2 \times \sigma^2)$. S_f is the sensitivity of f_{qr}, which means the maximum difference between query results on adjacent datasets.

Based on the above design, we discuss two types of interactions.

Task Offloading Between the Edge and the End. Considering the medical scenario, some internal staff know the disease diagnosis result but also leak the patient's physiological data from the hospital private server for interest. Thus, the edge server could also be honest-but-curious, that is, they honestly assist train the model, but meanwhile they want to infer or even construct user data information from the intermediate features [16], [37]. For the sake of security, the end perturbs O before sending out. Specially, we count the maximum and minimum value of each activation unit and compute the difference between the two values, and calculate their difference as S_f. Based on the above definition, we build function DP_1, as shown in (2). σ_e denotes

TABLE II
MAIN NOTATION DESCRIPTIONS IN OUR SCHEME

Notation	Description
τ	# iterations between adjacent aggregations
K	Round of global aggregation
x_k	Variable on whether to offload in round k
$< c_{off}, b_{off} >$	Computation resource cost for one iteration and communication cost for one round when $x_k = 1$
$< c_{loc}, b_{loc} >$	Computation resource cost for one iteration and communication cost for one round when $x_k = 0$
$< c_i, b_i >$	Computation and communication resource cost of device i estimated in real time
R	Total resource budget
$\bar{R}$	Resource budget expenditure
$< \epsilon_e, \delta_e, \sigma_e >$	Privacy budget, possibility of privacy broken and noise intensity during task offloading
$< \epsilon_c, \delta_c, \sigma_c >$	Privacy budget, possibility of privacy broken and noise intensity during parameter uploading
b	Batch size
D_i	Dataset of device i
s	Device sampling rate
$< \gamma, \gamma' >$	Searching step of s and σ_e, respectively
H	Function to calculate the privacy budget expenditure
$w_i(t)$	Model parameters of device i in iteration t
$w(t)$	Global model parameters in iteration t
w^*	Optimal model parameters
η	Learning rate
φ	Constant control parameter
ρ	Lipschitz parameter of any $f_i(w)$ and $F(w)$
β	Smoothness parameter of any $f_i(w)$ and $F(w)$
f_i	Loss function in device i
F	Global loss function

the noise intensity to the edge server.

$$DP_1(O, \sigma_e) = O + N(0, S_f^2 \sigma_e^2) \tag{2}$$

Transmitting model updates between the network edge (including the edge and the end) and the cloud. w_i^j denotes the model updates of j_{th} network layer in the end i. When updating it to the cloud, we preprocess w_i^j, that is, perform L2-normalization to each w_i^j and scale it to an empirical value ζ. We set ζ as the sensitivity in DP_2, and function DP_2 is defined in (3). σ_c denotes the noise intensity to the cloud.

$$DP_2(w_i^j) = \zeta \times w_i^j / \| w_i^j \| + N(0, \zeta^2 \sigma_c^2) \tag{3}$$

IV. ADAPTIVE CONTROL ALGORITHM UNDER MULTI-CONSTRAINTS

Efficiency and data privacy of FL are actually non-orthogonal. In this section, we formulate their inter-correlation towards high inference accuracy, and present our adaptive control algorithm to the multi-constraint problem.

A. Problem Formulation

We first introduce the two main constraints, and then give the optimization goal. Table II lists the main notations.

Resource Budget. 'resource' is a generic concept, which could be computation or communication resources involved or required in the task, for example, time and energy. The resource budget is the upper limit of the sum of affordable resource cost of the end device during participating in FL.

We introduce $X = \{x_1, x_2 \dots x_K\}$[1] indicating the offloading decision. If $x_k = 1$, it denotes that the model is offloaded in round k. And it consumes c_{off} units of resources for computation in each training iteration and b_{off} for communication in each aggregation. Otherwise, it denotes the end trains all network layers at local. It consumes c_{loc} and b_{loc} for computation and communication. Thus, the resource constraint is illustrated as

$$\sum_{k=1}^{K} x_k (c_{off} \cdot \tau + b_{off}) + (1 - x_k)(c_{loc} \cdot \tau + b_{loc}) \leq R \tag{4}$$

Privacy Budget. We define that the privacy budgets of the edge server and the cloud are ϵ_e and ϵ_c, respectively.[2] The end device sets the values of ϵ_e and ϵ_c based on its privacy preference before participating in FL. According to the boosting theorem in [12], to guarantee the (ϵ, δ)-DP, it follows $\frac{4}{5} exp(-(\sigma\epsilon)^2/2) \leq \delta$. After i iterations, the guarantee would get weakened to $(H(\epsilon, p_b, i, \delta'), p_b i\delta + \delta')$-DP, where $\delta' > 0$, function H is described as

$$H(\epsilon, p_b, i, \delta') = p_b \epsilon \sqrt{2i \cdot ln(1/\delta')} + p_b i \epsilon (exp(\epsilon) - 1) \tag{5}$$

According to specific interaction scenarios in Section III-C2, p_b and i in the theorem have corresponding meanings. When analyzing privacy during offloading, p_b is the fraction of the batch size b to the dataset size, i is the training iterations τ. While analyzing privacy in model aggregation, p_b is the sampling rate, i is the aggregation rounds K. For other relevant parameters $(\sigma, \epsilon, \delta, \delta')$ in DP, we use subscript abbreviations to distinguish (i.e., cloud and edge). ϵ_0 is the initial value. Thus, the privacy constraint is illustrated as

$$H(\epsilon_0, b/|D_i|, \tau, \delta_e') \leq \epsilon_e \tag{6}$$

$$H(\epsilon_0, s, K, \delta_c') \leq \epsilon_c \tag{7}$$

Optimization Goal. To obtain a global model with high accuracy, we design to minimize the value of global loss function $F(w)$ under the above constraints. The definition of $F(w)$ is illustrated in (8). $D = \{D_1, D_2 \dots D_N\}$. N is the number of end devices.

$$F(w) = \frac{\sum_i^N |D_i| f_i(w)}{|D|} \tag{8}$$

We denote w^* as the optimized parameters to minimize the value of $F(w)$, and denote $w(T)$ as the global parameters in iteration T. We aim to minimize their gap. And we refer to the approximation of the gap in [8], which is shown in (9). η is the learning rate, φ is a constant control parameter, ρ is the Lipschitz parameter of any $f_i(w)$ and $F(w)$. And $h(\tau)$ is a function describing the gap between the parameters obtained

1.For generality and interpretation, we use a fixed state of model splitting to illustrate, that is, splitting the model into two fixed parts. Thus, x_k is binary indicating whether to offload the deeper part.

2.The end (e.g., medical wearable devices) faces different security threats from two different entities (e.g., medical institutions and intelligent medical alliances). In this paper, we assume that the two don't collude and the end has set up two privacy budgets for them.

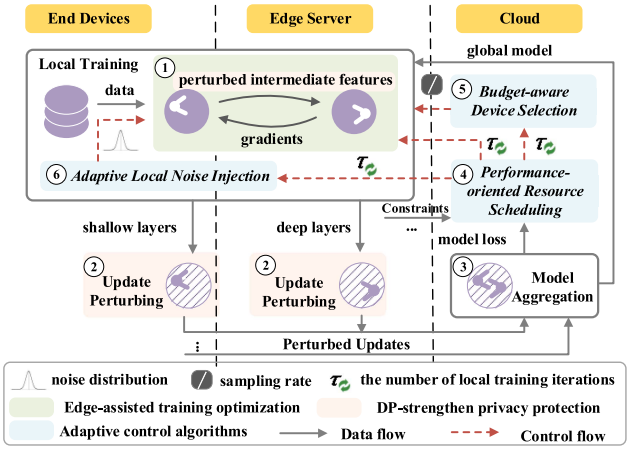


Fig. 3. Workflow of AHFL under both resource requirement and privacy constraints.

from distributed and centralized training. The definition is given in (10), where β is the smoothness parameter of any $f_i(w)$ and $F(w)$, and μ is the weighted average of $\{\mu_1, \mu_2, \ldots \mu_N\}$ based on their dataset size. Each μ_i follows (11).

$$F(w(T)) - F(w^*) \leq \frac{1}{2\eta\varphi T} + \sqrt{\frac{1}{4\eta^2\varphi^2 T^2} + \frac{\rho h(\tau)}{\eta\varphi\tau}} + \rho h(\tau) \tag{9}$$

$$h(\tau) = \frac{\mu}{\beta}((\eta\beta + 1)^\tau - 1) - \eta\mu\tau \tag{10}$$

$$\| \bigtriangledown f_i(w) - \bigtriangledown F(w) \| \leq \mu_i \tag{11}$$

Note that on the basis of the approximation of optimization goal, we develop it with more careful designs on network edge resource managing and privacy concerns. It strikes a more challenging balance among efficiency, accuracy and privacy in FL since the early depletion of any budget will lead to the underutilization of another budget and the early termination of training. In summary, our problem can be formulated as (12). To avoid excessive parameter adjustment, we empirically set σ_e, s, X, τ as adjustable variables and others as given constants.

$$\min_{\tau, X, \sigma_e, s} \quad \frac{1}{2\eta\varphi T} + \sqrt{\frac{1}{4\eta^2\varphi^2 T^2} + \frac{\rho h(\tau)}{\eta\varphi\tau}} + \rho h(\tau)$$
$$s.t. \quad (4), (6), (7) \tag{12}$$

B. Algorithm Design

In this subsection, we design AHFL, which accommodates both the resource and privacy budgets by embedding the following three adaptive control algorithms (step 4-6 in Fig. 3) into the hierarchical framework (step 1-3 in Fig. 3).

1) Performance-Oriented Resource Scheduling: To improve the model performance, the cloud monitors the training state and resource consumption of devices, and optimizes the resource scheduling for FL tasks. The algorithm is shown in Algorithm 1.

Each device monitors its computation and communication resource cost to estimate the value of c_i and b_i. When device i offloads the task to the edge server, c_i and b_i are essentially estimated as c_{off} and b_{off}. Otherwise, they are estimated as c_{loc} and b_{loc}. And the device reflects the distributed training state via estimated ρ_i, β_i, $f_i(w(t))$, $\bigtriangledown f_i(w(t))$ when receiving the global $w(t)$. These information is shared with the cloud to obtain $F(w(t))$, the weighted average of $f_i(w(t))$ according to (8) (line 1). The cloud estimates the global ρ, β, $\bigtriangledown F(w(t))$ and μ via averaging ρ_i, β_i and $\bigtriangledown f_i(w(t))$ weighted by $|D_i|/|D|$, respectively (line 2). Then it estimates μ_i based on (11) and calculates the global μ (line 3-4). The cloud feeds these parameters into the optimization function in (12), and finds the optimized value of τ within the interval $[1, \tau_{max}]$ via linear search (line 5). More details of the scheduling optimality are in Appendix IV-B1, available online. τ_{max} is the preset upper bound of τ. After that, the cloud checks the legality of τ in case of the exceed of the budget R (line 6-8). $\bar{R}$ records the resource budget already spent. At last, it returns τ for next round (line 9).

Algorithm 1: Performance-Oriented Resource Scheduling.

Input: R, $\bar{R}$, D_i, D, τ_{max}, ρ_i, β_i, $f_i(w(t))$, $\bigtriangledown f_i(w(t))$, c_i, b_i
Output: τ in next round
1: Calculate $F(w(t))$ based on (8);
2: Calculate the weighted average ρ, β and $\bigtriangledown F(w(t))$;
3: Estimate μ_i for each device i based on (11);
4: Calculate the weighted average μ;
5: Compute the optimized value of τ in (12) by linear search of integers within $[1, \tau_{max}]$;
6: **while** $\bar{R} + c_i\tau + b_i > R$ and $\tau > 0$ **do**
7: $\quad \tau \leftarrow \tau - 1$;
8: **end while**
9: **return** τ;

2) Budget-Aware Device Selection: We then integrate privacy concerns to the cloud into the above resource scheduling optimization. Small τ indicates frequent aggregation, which enlarges security risks from the cloud. We dynamically adjust the device sampling rate with awareness of the resource budget and privacy budgets of the cloud. The details are shown in Algorithm 2.

First, the cloud accumulates the current spent privacy budget $\bar{\epsilon}_0$ in this round according to function H (line 1). Then it estimates how many rounds can be executed with the new τ, which will consume the remaining resource budget (line 2). It searches for the optimal s via linear search with search step γ. The lower bound of s is s_{min}. The optimal s is supposed to make the spent budget $\bar{R}$ and $\bar{\epsilon}_0$ reach the total budget R and ϵ_c respectively at almost the same time, in case of early termination caused by lack of a certain type of budget (line 4-11). At last, s is returned as the next sampling rate.

3) Adaptive Local Noise Injection: We further design an adaptive local noise injection mechanism to minimize the end's resource consumption under the privacy constraint on the edge. High τ indicates frequent task offloading, which may easily break the constraint. Thus, before local training, the device

Algorithm 2: Budget-Aware Device Selection.

Input: s, s_{min}, ϵ_c, σ_c, δ_c, γ, R, $\bar{R}$, τ, c_i and b_i
Output: s in next round
1: Accumulate $\bar{\epsilon}_0$ with $\{\sigma_c, s, 1, \delta_c\}$ based on (5);
2: $\bar{k} \leftarrow (R - \bar{R})/(c_i\tau + b_i)$; // estimated remaining rounds
3: Initialize $\bar{\epsilon}_{tmp} \leftarrow \bar{\epsilon}_0$;
4: **while not** ($\bar{\epsilon}_{tmp} \leq \epsilon_c$ **or** $s < s_{min}$) **do**
5: $s \leftarrow s - \gamma$, $\bar{\epsilon}_{tmp} \leftarrow \bar{\epsilon}_0$;
6: Accumulate $\bar{\epsilon}_{tmp}$ with $\{\sigma_c, s, \bar{k}, \delta_c\}$ based on (5);
7: **end while**
8: **while not** ($\bar{\epsilon}_{tmp} \geq \epsilon_c$ **or** $s == 1$) **do**
9: $s \leftarrow s + \gamma$, $\bar{\epsilon}_{tmp} \leftarrow \bar{\epsilon}_0$;
10: Accumulate $\bar{\epsilon}_{tmp}$ with $\{\sigma_c, s, \bar{k}, \delta_c\}$ based on (5);
11: **end while**
12: **return** s

Algorithm 3: Adaptive Local Noise Injection.

Input: $\sigma_{e(min)}$, $\sigma_{e(max)}$, ϵ_e, τ, b, δ_e and γ'
Output: x_k, σ_e
1: $\sigma_e \leftarrow \sigma_{e(min)}$, $x_k \leftarrow 1$;
2: Compute $\bar{\epsilon}_e$ with $\{b, \sigma_e, \delta_e, \tau\}$ based on (5);
3: **while** $\bar{\epsilon}_e > \epsilon_e$ **do**
4: **if** $\sigma_e > \sigma_{e(max)}$ **then**
5: $x_k \leftarrow 0$;
6: break;
7: **end if**
8: $\sigma_e = \sigma_e + \gamma'$;
9: Compute $\bar{\epsilon}_e$ with $\{b, \sigma_e, \delta_e, \tau\}$ based on (5);
10: **end while**
11: **return** x_k, σ_e

evaluates whether the task can be offloaded with a guarantee to satisfy the privacy budget ϵ_e by controlling the strength of injected noise σ_e. The algorithm is shown in Algorithm 3.

First, we set σ_e as the lower bound $\sigma_{e(min)}$ and initial the offloading decision x_k as 1 (line 1). Then we compute spent privacy budget $\bar{\epsilon}_e$ using the above function H with the received τ and the current corresponding local configurations (line 2). $\bar{\epsilon}_e$ may exceed the preset budget ϵ_e. In that case, it would decrease the value of σ_e by search step γ' constantly until it satisfies the privacy constraints (line 8-9). Since the excessive noise makes the model hard to converge, we set the highest threshold for σ_e. If there is no σ_e within the preset interval satisfying the privacy budget, x_k would be 0 (line 4-7). At last, x_k and σ_e are returned to determine whether to offload and the noise intensity if offloading.

C. Security Analysis

We apply a two-level DP-strengthen mechanism to perturb the model parameters and the intermediate features, under strict cloud and edge privacy budgets that abide by DP.

Specifically, the model aggregation procedure satisfies (ϵ_c, δ_c)-DP. In each aggregation round, the cloud randomly selects end devices for training at the sampling rate s, and devices perturb their model parameters with noise intensity σ_c. The maximum difference of model parameters between two devices remains no greater than the sensitivity ζ in DP_2. It strictly obeys the Gaussian mechanism to satisfy DP. Algorithm 2 estimates the spent privacy budget after $\bar{k}$ rounds according to the boosting theorem in [12]. It adjusts s for the future rounds to guarantee that the spent budget is smaller than ϵ_c. If no appropriate s, the FL process would be stopped. Thus, this procedure satisfies (ϵ_c, δ_c)-DP.

The task offloading procedure satisfies (ϵ_e, δ_e)-DP. In each local training iteration, the end device perturbs the intermediate features of each data batch with noise intensity σ_e. The maximum difference of intermediate features between two data samples remains no greater than the sensitivity S_f in DP_1. It also strictly obeys the Gaussian mechanism. Since each batch is independent and satisfies the same DP, based on the parallel composition theorem in [12], the entire dataset is protected by the same DP protection strength. The privacy budget is accumulated over τ iterations. Algorithm 3 adjusts σ_e to guarantee that the accumulated privacy budget is still smaller than ϵ_e. Thus, this procedure satisfies (ϵ_e, δ_e)-DP.

In summary, AHFL could handle the threat from the cloud introduced in the threat model in Section III-B and resist the curious edge server when utilizing the edge's resource for training efficiency. Yet, we emphasize that AHFL is not designed to handle, so would unfortunately fail in face of, more deliberate security attacks from a) the collusion of the edge server and the cloud, b) the malicious edge server that dishonestly feeds back training gradients, and c) the malicious cloud who does not comply with aggregation rules. This specifies the security boundary of AHFL under honest-but-curious entities.

V. Evaluation

In this section, we evaluate our system through extensive experimental analysis to answer the following questions:

RQ1: Does our hierarchical FL framework relieve the end's resource burden?

RQ2: Does the adaptive control mechanism adjust relevant settings under given resource budgets and privacy budgets to maximize the performance of the training model?

A. Experimental Settings

Experiment Platform. We implement a simple networked prototype system, as shown in Fig. 4. Raspberry Pies and a Personal Computer (PC) emulate the resource-limited mobile end device and the edge server, respectively. The Raspberry Pies are equipped with 4 Cortex-A72 64@1.5GHZ and 4GB memory. The PC is equipped with 4 Intel(R) Core(TM) i5-4590@3.30GHZ and 8GB memory. The cloud has 20 Intel(R) Xeon(R) E5-2660 v3@2.60GHz and 62GB memory. As for the network connection, the Pies and the PC are in the same local area network, and the network between the PC and the cloud is the wide area network.

Dataset and Model Setting. We use CIFAR-10 and the original MNIST dataset in our evaluation. CIFAR-10 contains images of ten categories of items. Each category contains 60,000 images,

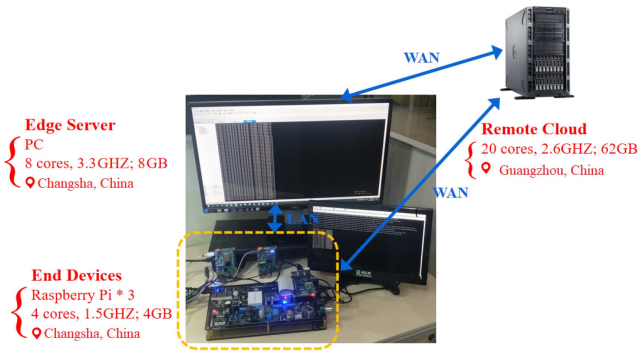


Fig. 4. Prototype of AHFL.

including 50,000 for training and the remaining 10,000 for testing. MNIST consists of 70,000 images of handwritten digits, of which 60,000 are for training and the remaining 10,000 are for testing. We simulate three different data distribution cases as follows. They reflect different degrees of Non-Independent Identically Distribution. And we use two neural network models[3] to train these datasets.

- *Case 1*: The training set is randomly and evenly distributed. Each end has uniform but partial information.
- *Case 2*: The training set is distributed based on the data labels. Each end has only partial data of a specific label.
- *Case 3*: The first half of the samples are randomly allocated based on the sample index as in Case 1, and the second half of the samples are randomly allocated based on the label as in Case 2.

Baseline. We compare AHFL with the following baselines.[4] The major differences reflect on the system structure, the flexibility of task schedule and the consideration of extra privacy risks of FL.

- CL [38], [39]: It refers to centralized learning. The cloud center collects data from the end devices and then trains a global model. It fails to provide any privacy protection.
- ltAdap [8]: It's general cloud-end FL with adaptable iterations between adjacent aggregations (i.e., adaptable τ). Yet, risks from the cloud are not considered. In each training round, every device will participate (i.e., without a selection process).
- FLGDP [17]: It provides membership inference resistance by implementing global DP on the aggregated model of general cloud-end FL. Here, the value of τ is fixed.
- FEEL [18]: It adopts a cloud-edge-end FL structure. It injects DP noise to both the model parameters in the cloud and the intermediate features. It fails to investigate the balance of privacy and efficiency by setting a fixed τ.

Parameter Setting. During local training, we set the batch size $b = 100$, the learning rate $\eta = 0.01$, the number of end devices $N = 30$. For the resource manager, we set the search step for sampling rate and noise intensity γ and γ' to 0.03 and 0.01, respectively. The constant control parameter φ is 5×10^{-5}. The upper limit of τ is 50.

B. Resource Consumption Analysis (RQ1)

We first measure and analyze the end device's resource consumption of our cloud-edge-end hierarchical FL framework. We compare it with the Traditional cloud-end FL framework (TFL). Both baseline ItAdap and FLGDP adopt such traditional framework. We divide the schemes adopting TFL into two categories according to whether they consider honest-but-curious cloud or not. They are abbreviated as TFL w/o privacy and TFL w/ privacy, respectively. All these schemes perform 10 epochs of local training and 1 round of global communication for aggregation. The results are illustrated in Table III.

The main difference between TFL w/privacy and TFL w/o privacy is reflected in the time consumption for the end's computation. Baseline TFL w/ privacy takes an extra 0.16s in CIFAR-10 and 0.12s in MNIST to perturb the local trained model parameters before sending them to the cloud. Owing to the richer edge resources, HFL decreases the time consumption for the end's computation by 50.5% than TFL w/ privacy. Although the end need to additionally perturb the intermediate features and model parameters, the time saved by model offloading training is higher than the consumption for perturbation. The main concern brought by offloading is the communication time between the end and the edge. We found that it takes 3.10s and 2.60s for such communication in the two datasets. Some compression methods could be introduced to reduce this consumption. The computation time cost in the edge is relatively small. In summary, the average local training time of HFL is decreased by 8.58%. HFL also makes use of the rich communication resource of the edge to upload the parameter of deep layers. It shortens the average global communication time by 59.35%. As for the memory consumption, HFL reduces it by 43.61% since it allows the end to only train a partial model. In summary, *our proposed HFL could reduce the end's resource consumption with the help of the edge.*

In the evaluation, w.l.o.g., we use time as the resource type. In this case, the corresponding computation resource cost is the time taken for local training and the communication resource cost is the time taken for exchanging model updates with the cloud. R is the upper limit of the total time cost of the end device in FL process.

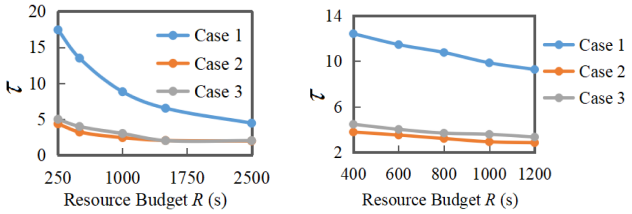
3.For CIFAR-10, the model has nine layers: $5 \times 5 \times 16$ Convolutional layer (CONV) $\rightarrow$ 2×2 MaxPool layer $\rightarrow$ Local Response Normalization layer (LRN) $\rightarrow$ $5 \times 5 \times 48$ CONV $\rightarrow$ LRN $\rightarrow$ 2×2 MaxPool layer $\rightarrow$ 3072×256 Dense layer $\rightarrow$ 256×10 Dense layer $\rightarrow$ Softmax layer. For MNIST, the model consists of nine layers: $5 \times 5 \times 16$ CONV $\rightarrow$ 2×2 MaxPool layer $\rightarrow$ LRN $\rightarrow$ $5 \times 5 \times 32$ CONV $\rightarrow$ LRN $\rightarrow$ 2×2 MaxPool layer $\rightarrow$ 2352×256 Dense layer $\rightarrow$ 256×10 Dense layer $\rightarrow$ Softmax layer. These mdoels' two convolutional layers and the first dense layer are followed by the activation function *ReLU*. Their loss functions are cross-entropy. They're compiled in Tensorflow.

4.We highlight that the comparison with ltAdap and FEEL also works as an ablation study for comprehensive evaluation of AHFL. Once the resource consumption runs out of budget, all these schemes would stop training. And if the current privacy budget has been exhausted but the resource budget has not been used up, these schemes would allow the ends to train local data independently without the corresponding cooperation with other entities.

TABLE III
RESOURCE CONSUMPTION W.R.T. TIME (LOCAL TRAINING AND GLOBAL COMMUNICATION) AND MEMORY

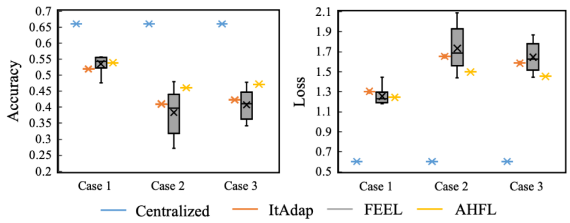
Dataset	Scheme	Local Training (s)			Global Comm. (s)	Memory Consumption (GB)
		End's Computation	End-edge Comm.	Edge's Computation		
CIFAR-10	TFL w/o privacy	9.50	/	/	5.00	3.3
	TFL w/ privacy	9.66	/	/	5.00	3.3
	HFL	5.13	3.10	0.41	2.02	1.6
MNIST	TFL w/o privacy	6.60	/	/	4.01	1.4
	TFL w/ privacy	6.72	/	/	4.01	1.4
	HFL	3.22	2.60	0.35	1.64	0.9

* Since baseline TFL w/o privacy and TFL w/ privacy don't involve the edge server in their frameworks, their end-edge communication time and edge's computation time are none.

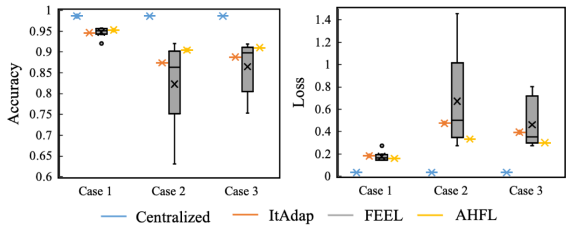


(a) CIFAR-10 (b) MNIST

Fig. 5. τ under different resource budget settings.



(a) CIFAR-10 (R=1500s)



(b) MNIST (R=800s)

Fig. 6. Comparison of accuracy and loss value under a given resource budget.

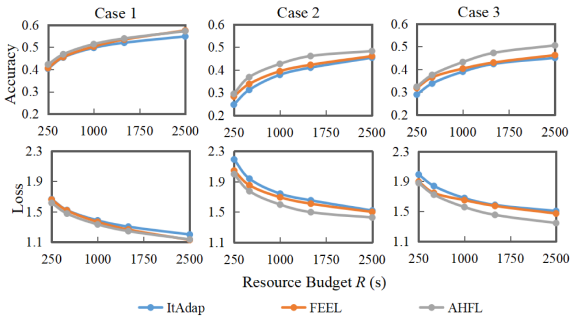
C. Effectiveness of the Adaptive Control Design (RQ2)

AHFL embeds three adaptive control modules based on HFL. In this subsection, we would verify the effectiveness of these three modules by measuring inference accuracy and loss value under limited resource and privacy budgets.

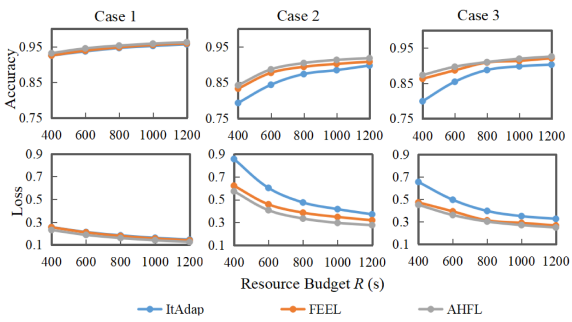
1) Resource Scheduling Module: Effectiveness. We adjust the total resource budget R to verify whether the resource scheduling module would perceive the resource budget and dynamically adjust the number of local training epochs τ in each round. Since the training complexity of the two data sets is different, we set different R for them. Specifically, for CIFAR-10, R is set from 250s to 2500s; for MNIST, R increases from 400s to 1200s.

We recorded τ in each round and calculated its average value. The results are shown in Fig. 5. We can observe that R indeed affects the resource scheduling of AHFL. With the increase of R, τ is gradually decreased, which means that more frequent parameter interactions are allowed to prevent the convergence direction of each end from being too scattered. Besides, the higher the degree of data imbalance, the lower τ. This is because the imbalance easily leads to great differences in the convergence direction.

Basic Performance. We then evaluate whether the dynamical τ in AHFL would benefit the training compared with other schemes. We set the resource budget R 1500s for CIFAR-10 and 800s for MNIST. Baseline FEEL adopts a fixed τ during FL training. Different τ makes FEEL perform differently, so we evaluate FEEL under a set of τ $\{1, 3, 5, 7, 10, 20, 30, 50\}$. Fig. 6 illustrates the comparison of accuracy and loss value among schemes under the given resource budgets. Baseline Centralized undoubtedly has the highest accuracy and lowest loss value regardless of the data distribution. But it fails to consider the data privacy. Baseline FEEL is not always superior to Baseline

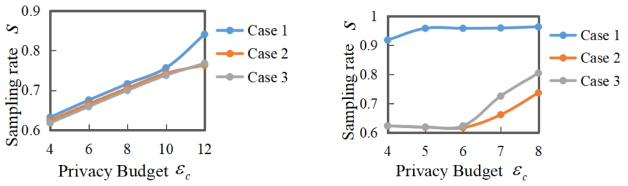


(a) CIFAR-10



(b) MNIST

Fig. 7. Accuracy and loss value under different resource budget settings.



(a) CIFAR-10 (R=1500s) (b) MNIST (R=800s)

Fig. 8. Sampling rate S under different privacy budget ϵ_c settings.

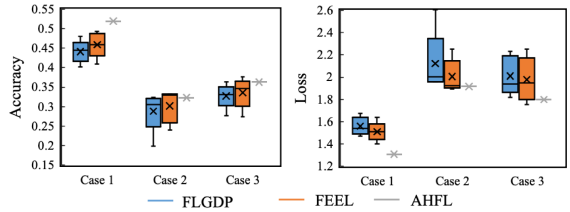
ItAdap. The reason is that although FEEL alleviates the impact of limited resource budget by offloading tasks, it neglects the importance of resource scheduling, resulting in it being inferior to Itadap in some cases of τ. But manually finding the best τ for FEEL requires much effort. Our scheme AHFL automatically schedules the resource while leveraging edge computing paradigm. The results demonstrate that *AHFL finds the near optimal resource scheduling and performs superior to ItAdap and FEEL in most cases.* Compared with ItAdap and FEEL, AHFL improves the average accuracy by 7.2% and reduces the average loss value by 17.5%.

Varying Resource Budgets. We further evaluate whether our scheme AHFL is still better than ItAdap and FEEL under different resource budgets. The results are shown in Fig. 7. Here we set τ fixed (10 for CIFAR-10 and 8 for MNIST) for FEEL. With these settings, FEEL performs well in the above basic performance evaluation. From Fig. 7, we found that AHFL can better adapt to the different R than the other two. In addition, we also find that the advantages are more prominent in the case of imbalanced data distribution and limited resource budget.

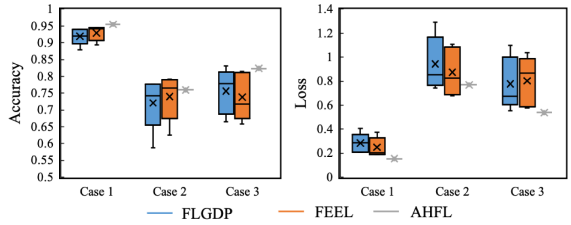
2) Device Selection Module: Effectiveness. We set multiple different privacy budgets of cloud ϵ_c to test whether the module would adjust the device sampling rate s accordingly. ϵ_c is set $\{4, 6, 8, 10, 12\}$ for CIFAR-10 and $\{4, 5, 6, 7, 8\}$ for MNIST. δ_c in DP is fixed at 1e-3. The resource budget R for the two datasets is set to 1500s and 800s respectively by default in subsequent experiments. Our scheme needs to accommodate both the efficiency expectation and privacy requirement. Here we keep the resource budget unchanged and increase the privacy budgets, simulating the transition of the main focus from privacy requirement to efficiency expectations.

We record s in each round and calculate its mean value. The results are shown in Fig. 8. It's found that s gradually increases with the growth of ϵ_c. The reason is that easing the privacy requirement against the cloud allows the devices to participate in FL more often to optimize the model.

Basic Performance. We then evaluate the performance with the adaptive device selection module under a given privacy budget of cloud. We set the budget ϵ_c 10 for CIFAR-10 and 6 for MNIST. We compare AHFL with Baseline FLGDP and FEEL because they consider the honest-but-curious cloud in FL. Considering the two fixed the device sampling rate s during FL training, we measure their performance when s is {0.2, 0.4, 0.6, 0.8, 1} respectively. Fig. 9 demonstrates that different s would lead to great differences in their performance. The reason is that high sampling rate s accelerates the consumption

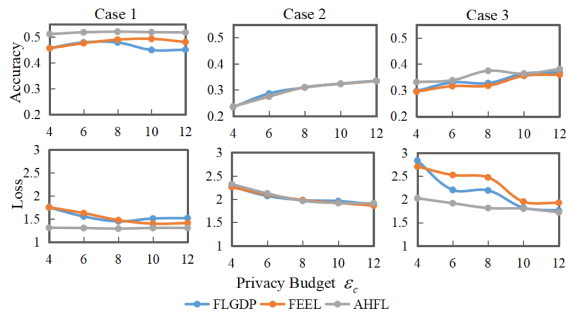


(a) CIFAR-10 (R=1500s, ϵ_c=10)

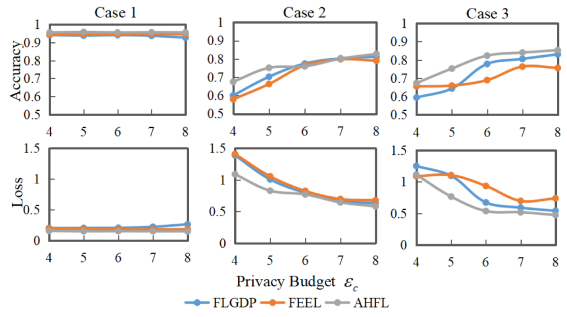


(b) MNIST (R=800s, ϵ_c=6)

Fig. 9. Comparison of accuracy and loss value under a given privacy budget of the cloud.



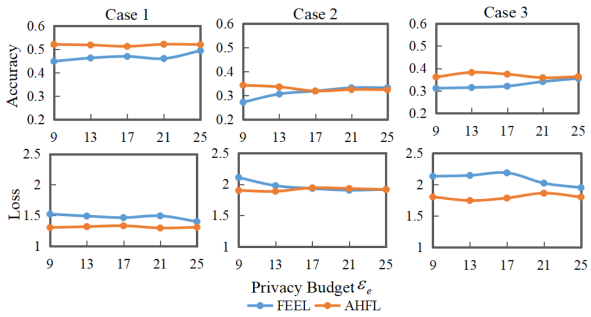
(a) CIFAR-10 (R=1500s)



(b) MNIST (R=800s)

Fig. 10. Accuracy and loss value under different privacy budget ϵ_c settings.

of the privacy budget and termination of FL, resulting in poor performance, while low s exhausts the resource budget without fully learning the knowledge of the end. The optimal s for different data distributions are usually different. For FLGDP, the optimal s is 1, 0.8 and 0.8 in Case 1-3 of CIFAR-10, 1, 0.6 and 0.6 in Case 1-3 of MNIST. Our scheme AHFL adjusts s without manual setting. From Fig. 9, we could observe that *the adjustment keeps the model performance at a high level in all these cases*. Compared with FLGDP and FEEL, the average accuracy of AHFL is improved by 8.67% and the average loss value is reduced by 20.1%.



(a) CIFAR-10 (R=1500s, ϵ_c=10)

(b) MNIST (R=800s, ϵ_c=6)

Fig. 11. Accuracy and loss value under different privacy budget ϵ_e settings.

Varying Privacy Budgets of the Cloud.

We further evaluate the impact of ϵ_c on the performance. The results are presented in Fig. 10. Considering that when s is set to 0.8, baseline FLGDP and FEEL perform well in the above basic performance tests. Here we set s 0.8 for them. Our scheme AHFL always performs better than these baselines under different ϵ_c, especially when ϵ_c is small. We also noticed that the performance improvement of AHFL is not very significant in some extreme cases. Specifically, in Case 2 of CIFAR-10, due to the high data imbalance and training complexity, the model convergence is difficult. In Case 1 of MNIST, owing to the uniform data distribution and low training difficulty, the model converges quickly even under stringent budgets. Thus, the performance of the three schemes is similar.

3) Local Noise Injection Module: We dynamically adjust the privacy budget of the edge ϵ_e to evaluate the effectiveness of this adaptive module. We compare AHFL with baseline FEEL, which also considers the malicious edge. FEEL calculates the minimum noise intensity σ_e satisfying (6) with the given ϵ_e, its fixed τ and $\delta_e = 1e-2$. σ_e is used to control noise injection during edge-end interaction. FEEL keeps offloading tasks until any budget are exhausted. Our scheme AHFL adaptively adjusts σ_e and offloading decision according to τ obtained in the resource scheduling module. Their performance comparison is demonstrated in Fig. 11. The average accuracy of AHFL is increased by 7.51% and the average loss value is decreased by 13.24%. As ϵ_e decreases, their performance differences always increase gradually. The reason is that FEEL insists on increasing the noise intensity σ_e to meet the privacy requirements during offloading tasks. However, excessive noise results in poor accuracy, which is greater than the advantages of offloading tasks. *AHFL can reasonably plan the privacy budget of the edge by dynamically adjusting noise injection and offloading decisions.*

VI. Conclusion

In this paper, we propose an adaptive hierarchical FL system to maximize inference accuracy under both resource and privacy budgets. The system involves edge computing paradigm to improve the training efficiency and applies DP mechanism to enhance the privacy of our design. An adaptive control algorithm is embedded in the system to coordinate the optimization of resource scheduling and the privacy protection operations to optimize the model performance. Extensive experiment results demonstrate that our scheme always achieves the best accuracy than other schemes. A key insight from our work is that the involvement of networked computation resources is expected to effectively relieve the tension between resource and privacy in distributed computing systems. With the large-scale deployments of 5G networks, we believe that the integration of FL, edge computing, and adaptable security primitives, will nurture the field for building more secure and efficient distributed computing architectures.

References

[1] E. Khramtsova, C. Hammerschmidt, S. Lagraa, and R. State, "Federated learning for cyber security: SOC collaboration for malicious URL detection," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 1316–1321.

[2] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 59–71, Jan. 2021.

[3] J. Guo, J. Wu, A. Liu, and N. Xiong, "LightFed: An efficient and secure federated edge learning system on model splitting," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 2701–2713, Nov. 2022.

[4] L. Li, H. Xiong, Z. Guo, J. Wang, and C. Xu, "SmartPC: Hierarchical pace control in real-time federated learning system," in *Proc. IEEE Real-Time Syst. Symp.*, 2019, pp. 406–418.

[5] A. C. Zhou, Y. Xiao, Y. Gong, B. He, J. Zhai, and R. Mao, "Privacy regulation aware process mapping in geo-distributed cloud data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 8, pp. 1872–1888, Aug. 2019.

[6] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 739–753.

[7] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[8] S. Wang et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[9] R. Chen, L. Li, K. Xue, C. Zhang, L. Liu, and M. Pan, "To talk or to work: Energy efficient federated learning over mobile devices via the weight quantization and 5g transmission co-design," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

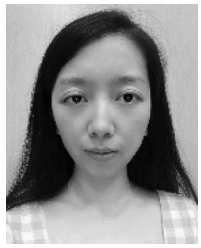
[10] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for D2D-enabled partial computation offloading in mobile edge computing," *IEEE Trans. Veh. Technol*, vol. 69, no. 4, pp. 4472–4486, Apr. 2020.

[11] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, 2021, Art. no. 94.

[12] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3/4, pp. 211–407, 2014.

[13] Y. Wang, S. Lu, and L. Zhang, "Searching privately by imperceptible lying: A novel private hashing method with differential privacy," in *Proc. 28th ACM Int. Conf. Multimedia*, 2020, pp. 2700–2709.

[14] H. Wang, G. Tang, K. Wu, and J. Wang, "PLVER: Joint stable allocation and content replication for edge-assisted live video delivery," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 1, pp. 218–230, Jan. 2022.
[15] S. Dong, D. Zeng, L. Gu, and S. Guo, "Offloading federated learning task to edge computing with trust execution environment," in *Proc. IEEE 17th Int. Conf. Mobile Ad Hoc Sensor Syst.*, 2020, pp. 491–496.
[16] Y. Mao, S. Yi, Q. Li, J. Feng, F. Xu, and S. Zhong, "Learning from differentially private neural activations with edge computing," in *Proc. IEEE/ACM Symp. Edge Comput.*, 2018, pp. 90–102.
[17] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*.
[18] Y. Guo, F. Liu, Z. Cai, L. Chen, and N. Xiao, "FEEL: A federated edge learning system for efficient and privacy-preserving mobile healthcare," in *Proc. 49th Int. Conf. Parallel Process.*, 2020, Art. no. 9.
[19] K. Bonawitz et al., "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*.
[20] B. Y. Lin et al., "FedNLP: A research platform for federated learning in natural language processing," 2021, *arXiv:2104.08815*.
[21] X. Huang, Y. Ding, Z. L. Jiang, S. Qi, X. Wang, and Q. Liao, "DP-FL: A novel differentially private federated learning framework for the unbalanced data," *World Wide Web*, vol. 23, no. 4, pp. 2529–2545, 2020.
[22] C. Niu et al., "Billion-scale federated learning on mobile clients: A submodel design with tunable privacy," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, Art. no. 31.
[23] Z. Jia and S. A. Jafar, "X-secure T-private federated submodel learning," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.
[24] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou, "A survey on edge computing systems and tools," *Proc. IEEE*, vol. 107, no. 8, pp. 1537–1562, Aug. 2019.
[25] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surveys*, vol. 52, no. 6, pp. 125:1–125:36, 2020.
[26] Y. Jin, L. Jiao, Z. Qian, S. Zhang, S. Lu, and X. Wang, "Resource-efficient and convergence-preserving online participant selection in federated learning," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 606–616.
[27] X. Zhang, X. Zhu, J. Wang, H. Yan, H. Chen, and W. Bao, "Federated learning with adaptive communication compression under dynamic bandwidth and unreliable networks," *Inf. Sci.*, vol. 540, pp. 242–262, 2020.
[28] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," 2021, *arXiv:2007.01154*.
[29] X. Pan, W. Wang, X. Zhang, B. Li, J. Yi, and D. Song, "How you act tells a lot: Privacy-leaking attack on deep reinforcement learning," in *Proc. 18th Int. Conf. Auton. Agents Multiagent Syst.*, 2019, pp. 368–376.
[30] N. Carlini et al., "Extracting training data from large language models," in *Proc. USENIX Secur. Symp.*, 2021, pp. 2633–2650.
[31] M. Song et al., "Analyzing user-level privacy attack against federated learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2430–2444, Oct. 2020.
[32] R. Kanagavelu et al., "Two-phase multi-party computation enabled privacy-preserving federated learning," in *Proc. IEEE/ACM 20th Int. Symp. Cluster Cloud Internet Comput.*, 2020, pp. 410–419.
[33] B. P. Knijnenburg and S. Berkovsky, "Privacy for recommender systems: Tutorial abstract," in *Proc. 11th ACM Conf. Recommender Syst.*, 2017, pp. 394–395.
[34] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong, "Personalized federated learning with differential privacy," *IEEE Internet of Things J.*, vol. 7, no. 10, pp. 9530–9539, Oct. 2020.
[35] Y. Chen, Y. Ping, Z. Zhang, B. Wang, and S. He, "Privacy-preserving image multi-classification deep learning model in robot system of industrial IoT," *Neural Comput. Appl.*, vol. 33, no. 10, pp. 4677–4694, 2021.
[36] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 683–697, Mar. 2022.
[37] S. Abuadbba et al., "Can we use split learning on 1D CNN models for privacy preserving training?," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2020, pp. 305–318.
[38] A. Salama, "Scalable data analytics and machine learning on the cloud," PhD dissertation, Dpt. Comput. Sci., Tech. Univ. Darmstadt, Darmstadt, Germany, 2021.
[39] M. Oh, S. Park, S. Kim, and H. Chae, "Erratum to: Machine learning-based analysis of multi-omics data on the cloud for investigating gene regulations," *Brief. Bioinf.*, vol. 22, no. 1, pp. 66–76, 2021.



Yeting Guo received the BS and MS degrees in computer science from the National University of Defense Technology, China, in 2017 and 2019, respectively. She is currently working toward the PhD degree with the College of Computer, National University of Defense Technology, Changsha, China. Her main research interests include computer architecture and edge computing.

Fang Liu received the BS and PhD degrees in computer science from the National University of Defense Technology, Changsha, China, in 1999 and 2005, respectively. She is a full professor with the School of Design, Hunan University. Her main research interests include edge computing, data storage and management, and Intelligent design.

Tongqing Zhou received the bachelor's, master's, and PhD degrees in computer science and technology from the National University of Defense Technology (NUDT), Changsha, China, in 2012, 2014, and 2018, respectively. He is currently a postdoctoral fellow with the College of Computer, NUDT. His main research interests include ubiquitous computing, mobile sensing, and data privacy.

Zhiping Cai received the BEng, MASc, and PhD degrees in computer science and technology from the National University of Defense Technology (NUDT), China, in 1996, 2002, and 2005, respectively. He is a full professor with the College of Computer, NUDT. His current research interests include artificial intelligence, network security, and big data. He is a distinguished member of the CCF.

Nong Xiao received the BS and PhD degrees in computer science from the College of Computer, National University of Defense Technology (NUDT), China, in 1990 and 1996, respectively. He is currently a professor with the School of Data and Computer Science, Sun Yat-sen University. His current research interests include network parallel computing, large-scale storage system, and computer architecture.