Accelerating Federated Learning Over Reliability-Agnostic Clients in Mobile Edge Computing Systems

Wentai Wu, *Student Member, IEEE*, Ligang He, *Member, IEEE*, Weiwei Lin, and Rui Mao

Abstract—Mobile Edge Computing (MEC), which incorporates the Cloud, edge nodes, and end devices, has shown great potential in bringing data processing closer to the data sources. Meanwhile, Federated learning (FL) has emerged as a promising privacy-preserving approach to facilitating AI applications. However, it remains a big challenge to optimize the efficiency and effectiveness of FL when it is integrated with the MEC architecture. Moreover, the unreliable nature (e.g., stragglers and intermittent drop-out) of end devices significantly slows down the FL process and affects the global model's quality in such circumstances. In this article, a multi-layer federated learning protocol called HybridFL is designed for the MEC architecture. HybridFL adopts two levels (the edge level and the cloud level) of model aggregation enacting different aggregation strategies. Moreover, in order to mitigate stragglers and end device drop-out, we introduce regional slack factors into the stage of client selection performed at the edge nodes using a probabilistic approach without identifying or probing the state of end devices (whose reliability is agnostic). We demonstrate the effectiveness of our method in modulating the proportion of clients selected and present the convergence analysis for our protocol. We have conducted extensive experiments with machine learning tasks in different scales of MEC system. The results show that HybridFL improves the FL training process significantly in terms of shortening the federated round length, speeding up the global model's convergence (by up to 12×) and reducing end device energy consumption (by up to 58 percent).

Index Terms—Federated learning, mobile edge computing, distributed computing, machine learning

1 Introduction

The rapid advance and remarkable achievements made in the development of Artificial Intelligence (AI) have drawn an unprecedented level of attention and revealed the potential of machine learning techniques. Meanwhile, the prevalence of Internet of Things (IoT) and Edge Intelligence [1] stimulates the efforts of pushing the computation to the edge of the network (closer to where the source of data resides) for faster response and better service quality [2]. With these two streams of research endeavour, it has been a major trend to empower the end devices in IoT with AI applications – Gartner has predicted that over 80 percent of enterprise IoT projects will incorporate AI components by 2022 [3]. Mobile Edge Computing (MEC) [4], [5], which consists of Cloud, edge nodes and end devices, is an emerging technology that can serve as the fundamental architecture of IoT, and provides a promising architecture for sinking AI to the edge nodes [6]. However, there are still many obstacles when it comes to the practical AI scenarios where the participants of the model training process are end devices such as cell phones, smart sensors and wearable electronics. First, although much work has shown good performance when training AI models in a cloud-centric manner using high-spec servers that hold the entire data set, it may not be feasible in many application scenarios (e.g., clinical diagnosis [7]) nowadays due to the data privacy concerns or the administration policies that forbid moving data out of local devices. Besides, though traditional distributed machine learning techniques (e.g., [8],[9]) can deal with decentralized data, they require very frequent exchange of gradients and model parameters, which results in heavy network traffic and prohibitive cost of communication in the cases where the devices are connected to the cloud via wireless channels.

Federated Learning [10], originally proposed by Google, is a distributed machine learning protocol designed for addressing the above-mentioned problems of data privacy and communication efficiency when training from decentralized data. A typical FL process consists of multiple rounds of training, in each of which clients (i.e., end devices) perform model training on local data and the cloud aggregates local models to produce a global model using a weight-averaging algorithm called *FedAvg*. As stated by McMahan *et al.* [10], the key properties of FL are: i) *unbalanced data distribution*: end devices may possess variable amounts of Non-IID (Non Independent and Identically Distributed) data; ii) *massively distributed devices*: the participants can be a huge fleet of heterogeneous end devices; iii) *limited access and communication*: data access is limited to

- *Wentai Wu, Ligang He are with the Department of Computer Science, University of Warwick, CV4 7AL Coventry, U.K. E-mail: {wentai.wu, ligang.he}@warwick.ac.uk.*
- *Weiwei Lin is with the School of Computer Science and Engineering at the South China University of Technology, Guangzhou, Guangdong Province 510641, China. E-mail: linww@scut.edu.cn.*
- *Rui Mao is with College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, Guangdong Province 518060, China. E-mail: mao@szu.edu.cn.*

Manuscript received 1 July 2020; revised 22 Aug. 2020; accepted 6 Sept. 2020. Date of publication 26 Nov. 2020; date of current version 11 Feb. 2021. (Corresponding author: Ligang He.) Recommended for acceptance by P. Balaji, J. Zhai, and M. Si. Digital Object Identifier no. 10.1109/TPDS.2020.3040867

1045-9219 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

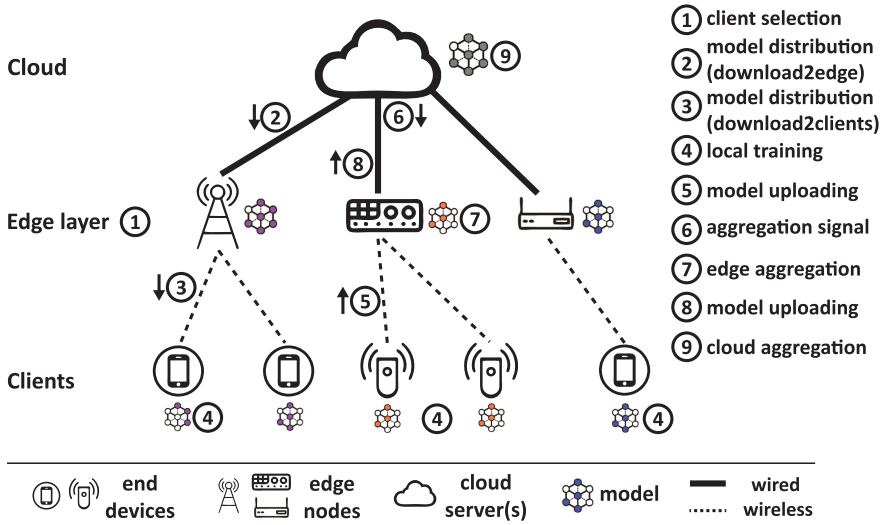


Fig. 1. A schematic overview of the HybridFL protocol designed for the MEC architecture, which consists of the Cloud, Edge layer, and Clients (end devices). The two levels of model aggregation are controlled by the "aggregation signal" sent from the cloud based on the information of local models collected by the edge nodes. In the system, clients are connected to the edge nodes via relatively low-speed (compared with high-speed network), shared (therefore noisy) wireless channels, whilst the edge-cloud connection is more stable (typically through Ethernet) and the bandwidth is typically sufficient.

local devices while the communication between the cloud and end devices can be slow and expensive.

Considering the privacy-preserving and communication-efficient nature of federated learning, it is now regarded as a promising approach to realizing intelligence on local devices and is a natural fit for the classic two-layer architectures such as cloud computing. However, with the emergence of MEC, there is still much room to explore in adapting the original FL protocol to the three-layer hierarchy of MEC, where abundant resources (in terms of computation, storage and network) are available in the edge layer. The necessity of such adaptation is two-fold. First, the network connection between the central server (i.e., the cloud) and the geographically distributed devices can be fairly slow and unreliable. Second, the central server may get overwhelmed by the workload or the network traffic due to excessive model updates from a vast number of clients. Introducing the edge layer, comprised of edge nodes such as base stations and servers in micro data centers, to the FL process can address these issues effectively. Edge nodes are proximate to the end devices and able to provide more stable connections and sufficient resources in computation. Existing work on edge-based FL [12], [13] has made use of edge resources or realized multi-step aggregation. However, the following issues have not been resolved yet: i) the *unreliability* of end devices (and their connections) is not considered; ii) the inefficiency of FL caused by the *heterogeneity* in end devices (and their bandwidth) is not addressed; iii) the *capability of edge nodes* (especially in terms of network) is not fully exploited.

The situation becomes even more challenging when combined with the strong privacy-preserving scenario, in which the servers are restricted from probing the information of clients [23]. For example, in the keyboard input prediction task, the device-server communication is stateless – end devices can be invited to participate, but cannot be tracked, and their reliability (i.e., the chance that they drop/opt out during the training) is agnostic.

In this paper, we propose a novel protocol (HybridFL) to enable privacy-preserving, efficient federated learning under the MEC architecture. Fig. 1 shows an overview of our protocol. We leverage the capability of edge nodes to boost the efficiency of communication and adapt FL to the three-layer hierarchy of MEC by making the FL process a hybrid of client-edge and edge-cloud collaborations. We also take into account both the unreliability and heterogeneity of end devices under strong privacy conditions and adopt a pace steering mechanism which is a hybrid of synchronous (edge-cloud) and asynchronous (client-edge) communication. The main contributions of our work are outlined as follows:

- We propose a novel protocol (HybridFL) to drive the FL process in the three-layer architecture of MEC. HybridFL facilitates efficient model exchanges via the combination of quota-triggered regional aggregation (via the edge layer) and immediate cloud aggregation.
- We mitigate the impact of client drop-out by introducing a regional slack factor for each edge node (i.e., region) into the client selection step via a probabilistic estimation method under strong privacy-preserving conditions that cause clients' reliability to be agnostic.
- We introduce Effective Data Coverage (EDC) into the step of cloud-level model aggregation and present the convergence analysis for our protocol.
- We conducted extensive experiments with machine learning tasks (using two public data sets) to evaluate the performance of our HybridFL protocol. Experimental results demonstrate significant improvement in average round duration, global model's convergence speed and accuracy, and the energy consumption of end devices.

The rest of this paper is organized as follows: Section 2 discusses the relevant studies concerning FL and MEC. In Section 3, we detail the design of HybridFL and give

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

convergence analysis for our protocol. In Section 4, we present and discuss the experimental results. We conclude this paper in Section 5.

2 Related Work

Stochastic Gradient Descent (SGD) [14],[15] and its variations have been the de facto standards for training most of the modern machine learning models. The fundamentals of previous distributed learning methods were built on the exchange of gradients, no matter in a centralized or decentralized manner. Extensive studies have shown the effectiveness of distributed SGD [16], [17] for complex models such as Deep Neural Networks (DNN). However, most of these traditional methods are designed for or tested in the data-center type of environments, where the data can be accessed globally by all workers and the financial cost of the communications is hardly considered as an issue. However, the real-life scenarios of MEC are usually comprised of low-spec, unreliable end devices, geographically distributed edge nodes with moderate performance, and relatively low-speed, noisy communication between edge nodes and end devices via wireless channels.

Federated Learning (FL), in a large part, addresses the problems of data privacy and prohibitive communication cost in training a global model from decentralized data. FL was originally designed as a synchronous training protocol called FedAvg [10], which is a weight-averaging algorithm and aggregates the local models from end devices to produce a new global model each round. Model exchange, which is much less frequent (and thereby more communication-efficient) than the gradient exchange in traditional distributed SGD methods, is the outstanding feature of FL. Many Studies show that there is still potential in further reducing the communication cost of FL via model compression [11], setting adaptive aggregation intervals [12] and using multi-task learning [18]. In addition, many variants of the FL protocol have also been proposed. For example, Xie *et al.* [19] and Sprague *et al.* [20] adopted asynchronous federated optimization schemes with non-blocking global model update and allowing devices to join halfway during training. SAFA [21] is semi-asynchronous FL protocol that retains the synchronized pace steering while introducing the strategies such as post-training client selection and model caching to speed up the training process.

It is natural to adapt FL to the emerging Mobile Edge Computing. A number of studies have provided their solutions to such adaptation. Wang *et al.* [12] took into account the resource budgets for edge nodes and proposed a pace control algorithm that adaptively adjusts the aggregation interval of FL. They adopted a system architecture in which the data reside on edge nodes, which essentially makes the system still a two-layer FL protocol. Liu *et al.* [13] implemented a hierarchical FL protocol that utilizes the cloud server and the edge nodes to perform two levels of model aggregation. The protocol is a straightforward extension of FedAvg, allowing multiple rounds in the edge layer before a global aggregation by the cloud. A main problem of the work is that each pair of interactions (i.e., client-edge and edge-cloud) is tightly coupled. As a result, device/network failures will cause both the edge nodes and the cloud to wait for a long time. In addition, in HierFAVG, edge nodes have to perform multiple rounds of edge-level aggregation before sending models to the cloud. This significantly postpones the global exchange of model information and consequently slows down the convergence. Considering the drawbacks of the existing solutions in driving FL in the MEC systems, in this work we aim to develop a more efficient FL protocol that enables fast, robust machine learning by virtue of the resources in the edge layer.

3 The HybridFL Protocol

There are eight steps in each round of training using the HybridFL protocol to drive FL in the MEC architecture (see Fig. 1). These eight steps form three basic stages: *model distribution*, *local training* and *model aggregation*. The stage of model distribution starts with client selection (step 1, Fig. 1), after which the (latest) global model is distributed over the edge nodes (step 2) and then across all the clients (step 3). The second stage, local training, is performed on the clients (end devices) and also includes the steps of model downloading and uploading via the client-edge connections. Aggregation is the final stage of a round where the local models (selected and uploaded) are merged into a global model. Our protocol adopts a hybrid pace steering mechanism that allows flexible control over the edge-level (i.e, regional) model aggregation, which is signified by the cloud.

In this section we present the detailed design of HybridFL, in particular how we introduce regional slack factors into the model distribution stage and how our protocol performs the aggregations at the edge- and cloud-level.

In this paper, we refer to the collection of clients connected to an edge node as a *region*. D^r denotes the set of data in region r (note that the data cannot leave their end devices), i.e., $D^r = \{D_k^r | \forall k \in \text{region}\, r\}$. Without loss of generality, we assume a client can only connect to a single edge node. Table 1 lists the notations frequently used in this paper.

In the first stage of any FL round, client selection is often performed to ensure that only a reasonable proportion of clients are engaged in training this round. For example, the number of selected clients is determined by the proportion C in [10], [22]. As pointed out by Kairouz *et al.* [23], it is necessary to restrict the participating population to a small fraction for two reasons. First, it has been shown that involving an excessive number of clients can hardly benefit the convergence and quality of the global model [10]. Second, recruiting excessive devices is neither cost-efficient in communication nor realistic for the end device owners.

Nevertheless, a severe shortage of participants in FL also leads to an inferior global model, of which the unreliability of end devices is the main cause. These devices can opt out of any round of local training or drop out occasionally due to various reasons such as low battery level, device failure or network disconnection. Let $X_r(t)$ denote the set of clients who are in region r and do not drop out of round t. Then, since the client may drop-out, manually or unexpectedly, we have $|X_r(t)| \leq C \cdot n_r$ and $|X(t)| = \sum_{r=1}^{m} |X_r(t)| \leq C \cdot n$, where C is the desired proportion of clients with successful model submission (C is preset by the cloud server).

In order to mitigate the shortage of participants caused by drop-out, we introduce $C_r(t)$ as the region-wise selection

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
List of Symbols

Symbol	Description
D	the complete dataset
D_k^r	the data partition on client k in region r
D^r	the (logical) set of data in region r
n	the number of clients
n_r	the number of clients connected to edge node r
m	the number of edge nodes (regions)
V_C	the set of clients
V_E	the set of edge nodes
V_C^r	the set of clients in region r ($\|V_C^r\| = n_r$)
w	parameters of the global model
w^r	parameters of the model on edge node r
w_k^r	parameters of the local model on client k
C	the desired proportion of clients that submit their local models in a round. C is specified by the cloud.
C_r	the proportion of clients selected in region r
$U(t)$	the set of selected clients in round t
$U_r(t)$	the set of selected clients within region r in round t; $\|U_r(t)\| = C_r \cdot n_r$
$X(t)$	the set of all clients (across all regions) that do not drop out in round t
$X_r(t)$	the set of clients in $X(t)$ belonging to region r
$S(t)$	the set of clients that submit their models in time and successfully in round t
$S_r(t)$	the set of clients in $S(t)$ belonging to region r

proportion into the client selection step (i.e., step 1 in Fig. 1) of the HybridFL training process at the start of each round. More specifically, an edge node r will determine $C_r(t)$ and select a fraction of $C_r(t) \cdot n_r$ clients randomly (the set of selected clients is denoted by $U_r(t)$) before signifying these clients to begin local training in round t. An ideal value of $C_r(t)$ should satisfy that: i) the resulting $|U_r(t)|$ should be large enough so that the stragglers and dropouts have the minimal impact on round efficiency, and ii) $|U_r(t)|$ should not be too large, otherwise local training on some devices may be futile because the cloud only accepts a maximum of $C \cdot n$ clients each round. The main challenge here is that it is not permitted for an edge node to probe the state of its clients (including their IDs, aliveness, training progress, and the number of model updates made by a particular client), which causes the client's reliability (i.e., the probability that it drops out in a round) to be agnostic to the edge and the cloud. In view of this, we develop a probabilistic approach in this work to determine $C_r(t)$ for each region.

3.1 Regional Client Selection

By specifying the regional selection proportion $C_r(t)$ in round t, we aim to involve a fraction of $C_r(t)n_r$ clients in region r and expect that $C \cdot n_r$ of them do not drop/opt out, provided that all these clients may be unreliable. Formally, the target of our region-wise selection can be formulated as:

$$\mathbb{E}[|X_r(t)|; C_r^*(t), n_r] = C \cdot n_r, \tag{1}$$

where $\mathbb{E}[|X_r(t)|; C_r^*(t), n_r]$ is the expectation of the number of clients in region r that do not drop out in round t given that an optimal proportion of clients $C_r(t) = C_r^*(t)$ are selected from n_r (i.e., total number of clients in region r) clients to perform local training.

Given any selection proportion $C_r(t)$, the expectation at the left-hand-side of (1) is equivalent to:

$$\mathbb{E}[|X_r(t)|; C_r(t), n_r] = \sum_{k=0}^{C_r(t)n_r} k \sum_{b \in \mathrm{comb}(U_r(t),k)} P(b), \tag{2}$$

where $U_r(t)$ is the set of clients selected in region r, $comb(U, k)$ is the set of all combinations when selecting k elements from the set of U, and $P(b)$ is the probability that the combination b of end devices happen to be those who do not drop out in round t. Given a combination $b \in comb(U_r(t), k)$ and let $P_i^r(t)$ denote the probability that device i of region r does not drop/opt out in round t, $P(b)$ can be calculated by (3):

$$P(b) = \prod_{i \in b} P_i^r(t) \cdot \prod_{i \notin b} (1 - P_i^r(t)). \tag{3}$$

Therefore, to obtain the optimal client selection proportion $C_r^*(t)$ we must solve it from (4) combining (3):

$$\sum_{k=0}^{C_r^*(t)n_r} k \sum_{b \in \mathrm{comb}(U_r(t),k)} P(b) = C \cdot n_r. \tag{4}$$

However, $C_r^*(t)$ cannot be solved from (4) without a priori knowledge on the probability $P_i^r(t)$ (i.e., reliability) of every individual client. In this work we consider a FL scenario with strong privacy-preserving condition, under which it is prohibited to acquire the clients' identifiers and their states [23], i.e., $P_i^r(t)$ is agnostic. In view of this, we develop a novel approach to address this difficulty and eventually work out the optimal value of $C_r(t)$ for each region r each round t.

Assume that $\theta_r(t)$ is such a probability that after we replace each individual $P_i^r(t)$ in (3) with $\theta_r(t)$, the resulting expectation of $|X_r(t)|$ remains unchanged. We can always find such $\theta_r(t)$ because after the replacement, $|X_r(t)|$ of region r follows the Binomial distribution $\mathcal{B}(C_r(t)n_r, \theta_r(t))$ and the expectation of $|X_r(t)|$ (i.e., $\mathbb{E}[|X_r(t)|; C_r(t), n_r] \in [0, C_r(t)n_r]$) is a surjective function of $\theta_r(t) \in [0, 1]$. Now we can re-write the right-hand side of (2):

$$\begin{aligned} \mathbb{E}[|X_r(t)|; C_r(t), n_r] &= \sum_{k=0}^{C_r(t)n_r} k \cdot P(|X_r(t)| = k) \\ &= C_r(t) n_r \theta_r(t), \end{aligned} \tag{5}$$

where the second equality in (5) holds because $|X_r(t)| \sim \mathcal{B}(C_r(t)n_r, \theta_r(t))$.

Combining (5) and our selection target (1), we have:

$$C_r(t) = \frac{C}{\theta_r(t)} \text{ if } C_r(t) = C_r^*(t), \tag{6}$$

where C is the desired global proportion of clients (specified by the cloud) with successful model submissions in round t over the entire MEC system. $\theta_r(t)$ defined in (5) modulates the selection proportion in a region to compensate the client drop-out in that region. Therefore we term $\theta_r(t)$ the *regional slack factor* (for region r).

Note that $\theta_r(t)$ in (6) cannot be decided arbitrarily, otherwise the optimality of $C_r(t)$ is not guaranteed. This is

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

because there is only one optimal value for $C_r(t)$ given any distribution of client reliability and the target formulated in (1). In other words, if we determine $C_r(t)$ via (6) provided an under-estimated $\theta_r(t)$, the selection proportion $C_r(t)$ will be too big (i.e., $C_r(t) > C_r^*(t)$) for region r and consequently, the target expectation of $|X_r(t)|$ will be higher than the desired level, i.e., $\mathbb{E}[|X_r(t)|; C_r(t), n_r] > C \cdot n_r$. It is similar for the situation of over-estimated $\theta_r(t)$.

According to (6), we can determine how many clients we need to select in each region for a upcoming FL round after $\theta_r(t)$ is resolved. In this work, we develop a novel method to estimate $\theta_r(t)$ based on the historical records of model submissions (since the course of FL is organized in rounds), i.e., how many models are collected by each region in previous rounds. Note that edge nodes can only count the models they collected but do not know which client submitted the model.

In HybridFL, we adopt a quota-triggered aggregation mechanism in which the cloud ends a round once $C \cdot n$ client models have been submitted globally across the MEC system. As a result, we have:

$$\sum_{r \in V_E} |S_r(t)| = \min(nC, \sum_{r \in V_E} |X_r(t)|), \tag{7}$$

where $S_r(t)$ is the set of clients that submit their models in time in round t (before the cloud ends a round after collecting $C \cdot n$ models globally) and V_E is the set of edge nodes. Note that $S_r(t) \subseteq X_r(t)$ because when the cloud ends a round, some clients may be still working and have not finished local training. Details of how to determine the aggregation timing will be introduced later. Formally, we use a factor $q_r^*(t)$ to characterize the relation between $|S_r(t)|$ and $|X_r(t)|$:

$$|S_r(t)| = |X_r(t)| \cdot q_r^*(t), \tag{8}$$

where $q_r^*(t)$ denotes the percentage of clients in $X_r(t)$ that submit local models in time (these clients make up $S_r(t)$). Note that $|S_r(t)|$ is observable as the number of local models collected by edge node r in round t. However, $X_r(t)$ is agnostic since we consider a strong privacy-protection scenario where the edge nodes are not allowed to probe the state of clients. We can only observe how many clients submitted the updated models (i.e., $|S_r(t)|$) but cannot know who have dropped out and who are still working. Therefore, we transform (8) into (9) given $\mathbb{E}[|X_r(t)|; C_r(t), n_r] \neq 0$, and then define $q_r(t)$ in (10).

$$|S_r(t)| = \mathbb{E}[|X_r(t)|; C_r(t), n_r] \cdot \frac{|X_r(t)| \cdot q_r^*(t)}{\mathbb{E}[|X_r(t)|; C_r(t), n_r]} \tag{9}$$

$$q_r(t) \triangleq \frac{|X_r(t)| \cdot q_r^*(t)}{\mathbb{E}[|X_r(t)|; C_r(t), n_r]}. \tag{10}$$

From (9), (10) and (5), we have:

$$\begin{aligned} |S_r(t)| &= \mathbb{E}[|X_r(t)|; C_r(t), n_r] \cdot q_r(t) \\ &= C_r(t) n_r \theta_r(t) \cdot q_r(t). \end{aligned} \tag{11}$$

Note that the value of $\theta_r(t)$ needs to be estimated before round t starts so that we can determine the selection proportion $C_r(t)$ (every round begins with the client selection step). However, (11) cannot be used directly to obtain $\theta_r(t)$ because $S_r(t)$ and $q_r(t)$ are unknown before round t is completed (Note that $S_r(t)$ is observable at the end of round t so we can calculate $q_r(t)$ with $|S_r(t)|$ at round ends by (12) combining (10), (8) and (1) with the assumption that $C_r(t)$ is the optimal).

$$q_r(t) = \frac{|S_r(t)|}{C \cdot n_r}. \tag{12}$$

Therefore we develop the following practical approach to work out $\theta_r(t)$ by exploiting the historical records of the variables observable to edge nodes. More specifically, edge node r has stored $S_r(1), S_r(2), \ldots, S_r(t-1)$, $q_r(1), q_r(2), \ldots, q_r(t-1)$ and $C_r(1), C_r(2), \ldots, C_r(t-1)$ at the start of round t. Also, according to the definition of $\theta_r(t)$, it represents a region-wise property. So we assume $\theta_r(t)$ does not change significantly over the course of the FL training. Thus, we use a constant $\hat{\theta}_r(T)$ as the approximation of $\theta_r(t)$ within the time window T spanning from round 1 to round t:

$$\theta_r(i) \approx \hat{\theta}_r(T), \forall i \in \{1, 2, \ldots, t\}. \tag{13}$$

Replacing $\theta_r(i)$ with $\hat{\theta}_r(T)$ in (11) and for round i, $\forall i < t$, we have:

$$\frac{|S_r(i)|}{n_r} \approx C_r(i) q_r(i) \hat{\theta}_r(T), \forall i \in \{1, 2, \ldots, t-1\}. \tag{14}$$

Therefore, (14) is equivalent to a series of observations (the number of which is $t-1$) sampled from a function in the form of "$y = ax$" (i.e., $|S_r(i)|/n_r$ and $C_r(i)q_r(i)$ being the samples of y and x, respectively, and $\hat{\theta}_r(T)$ is the coefficient). In view of this, we use Least Square Estimation (LSE) to fit the value of $\hat{\theta}_r(T)$ based on (14), which produces:

$$\hat{\theta}_r(T) \overset{LSE}{=} \frac{1}{n_r} \frac{\sum_{i=1}^{t-1} C_r(i) q_r(i) |S_r(i)|}{\sum_{i=1}^{t-1} \left(C_r(i) q_r(i)\right)^2}, \ t > 1. \tag{15}$$

where $S_r(i)$, $C_r(i)$ and $q_r(i), i = 1, 2, \ldots, t-1$ are retrieved from the logs of edge node r. At the start of round t, we compute $\hat{\theta}_r(T)$ and use it as an estimate of $\theta_r(t)$, and then determine region r's client selection proportion $C_r(t)$ (defined in (6)) using (16):

$$C_r(t) = C \cdot n_r \frac{\sum_{i=1}^{t-1} \left(C_r(i) q_r(i)\right)^2}{\sum_{i=1}^{t-1} C_r(i) q_r(i) |S_r(i)|}, \ t > 1. \tag{16}$$

For t=1 (the 1st round of FL), $\theta_r(t)$ is initialized as a default value (e.g., $\theta_r(1)$=0.5). $C_r(1)$ is initialized to $C/\theta_r(1)$ accordingly. To investigate the effectiveness of our method in terms of achieving the selection target (1), we simulated 20 clients in two regions and ran 100 rounds (5 local epochs in each round) of federated learning using HybridFL as the control protocol. We initialized $\theta_r(1)$ to 0.5.

From the traces of $\theta_r(t)$, $C_r(t)$, $q_r(t)$ and $|X_r(t)|/n_r$ in Fig. 2, we can observe that our probabilistic estimation drives $\theta_r(t)$ and $C_r(t)$ (the first two rows in the figure) to the convergence at about 40 rounds of FL. Note that $\theta_1(t)$ and $\theta_2(t)$ converge to 0.46 and 0.63, which, by the definition, are

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

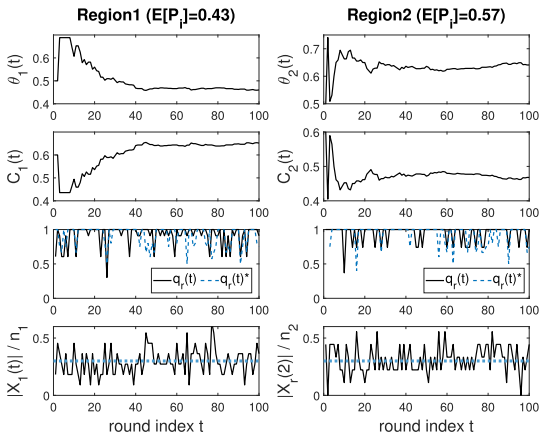


Fig. 2. The traces of $\theta_r(t)$, $C_r(t)$, $q_r(t)$ and $|X_r(t)|/n_r$ in a simulation where there are 11 and 9 clients in Region 1 and Region 2, respectively. The reliability of clients in training follows the Gaussian distribution $\mathcal{N}(\mu, 0.15^2)$ where $\mu = \mathbb{E}[P_i^r(t)]$ is set to 0.43 and 0.57 for Region 1 and Region 2, respectively. Clients also differ in performance which follows $\mathcal{N}(0.5, 0.1^2)$; The global selection fraction C is set to 0.3.

not necessarily equal to $\mathbb{E}[P_i^r(t)]$ (recall that $P_i^r(t)$ is the reliability of client i in region r) which is set to 0.43 and 0.57 for Region 1 and Region 2 in this example, respectively. Besides, we define $q_r(t)$ without using any knowledge about $X_r(t)$, but still produces a close approximation to its true value $q_r^*(t) = |S_r(t)|/|X_r(t)|$ (the 3rd row in Fig. 2). Consequently, the client participating ratio in a region, quantified by $|X_r(t)|/n_r$, is maintained around $C = 0.3$ (shown in the last row of Fig. 2; the blue dash line represents $C = 0.3$) after the convergence of $\theta_r(t)$ and $C_r(t)$.

With this case we demonstrate that our method for estimating $\theta_r(t)$ (which determines $C_r(t)$) is both theoretically and practically feasible for finding the optimal value of the regional selection proportion that leads to the very expectation of $|X_r(t)|$ desired by the cloud (see the target (1)).

3.2 Model Aggregations

In our protocol (HybridFL), model aggregation is a multi-step stage, involving both edge- and cloud-level aggregation (see steps 6, 7, 8 and 9 in Fig. 1). In HybridFL, once the updated models submitted by the clients across the MEC system equals to $C \cdot n$ (i.e., $|S(t)| = \sum_{r \in V_E} |S_r(t)|$ reaches $C \cdot n$), it triggers the cloud to send the "aggregation signal" to the edge nodes (see step 6, Fig. 1). The edge nodes will then stop waiting for more local models. This quota-triggered regional aggregation effectively mitigates the impact of the clients which have dropped out or are straggling. Consequently the round length is expected to be shortened (our experiment results support this expectation).

We adopt an *immediate cloud aggregation* strategy, which allows the cloud-level model aggregation to be conducted right after the edge-level aggregation is completed. The rationale behind this strategy is that the cloud-edge network connection is typically reliable and of low latency. Therefore, it facilitates the global information exchange and the convergence of the global model by aggregating the regional models at the cloud level as early as possible after the regional aggregations are completed at the edge nodes. Fig. 3 demonstrates how rounds are orchestrated in HybridFL.

The cloud keeps monitoring the total number of clients that have submitted their models each round by listening to

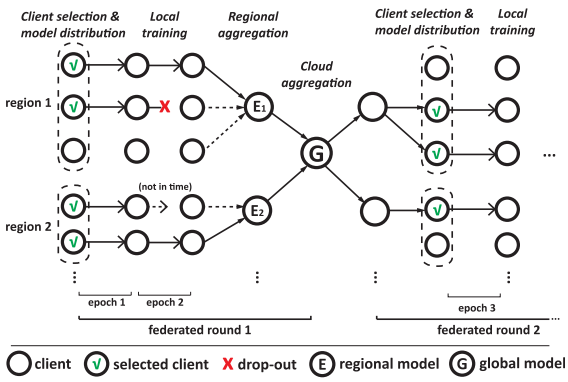


Fig. 3. The workflow of the proposed HybridFL protocol wherein the cloud requires two local model submissions (since C=0.4 and n=5 in this example) each round to trigger the model aggregation. Local training consists of two epochs. Note that the models from the dropped-out or straggling clients (e.g., clients no. 2 and 4 in round 1) are not actually uploaded (dashed line with arrow).

the reports of the current value of $|S_r(t)|$ from the edge nodes. Once the total number of client model submissions reaches the quota $C \cdot n$, the cloud will signify the edge nodes to perform regional aggregation, the result of which can be formulated as:

$$w^r(t) = \sum_{k \in V_C^r} \frac{|D_k^r|}{|D^r|} w_k^r(t), \tag{17}$$

where $w_k^r(t)$ is the model on client k in region r and $w^r(t)$ denotes the resulting regional model for edge node r in round t. Note that the aggregation involves all client models in the region, not limited to those who finished local training successfully (see Fig. 3). To alleviate model staleness, we use a cache solution in which the local models without successful update in the current round are replaced with the existing regional model obtained in last round before the aggregation is conducted, i.e., $w_k^r(t) = w^r(t-1)$ if $k \notin S_r(t)$.

The cloud aggregation will be performed immediately after the regional aggregation to produce the cloud model. Instead of using constant weight for each regional model as in the literature [13], we adopt a data-oriented weight averaging strategy by introducing the *Effective Data Coverage* (EDC) for each region in every round. EDC quantifies the actual size of data covered in round t's training based on $S_r(t)$. We formulate EDC for region r in round t (denoted by $EDC_r(t)$) as:

$$EDC_r(t) = \sum_{k \in S_r(t)} |D_k^r|, \tag{18}$$

where $S_r(t)$ is the set of clients who submitted their models successfully to its regional edge node. Accordingly, we further define EDC for the whole MEC system (denoted by $EDC(t)$) as:

$$EDC(t) = \sum_{r \in V_E} EDC_r(t). \tag{19}$$

In the model aggregation step at the cloud level, we weight each regional model $w^r(t)$ based on EDC to characterize its round-wise contribution in producing the global model $w(t)$:

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

$$w(t) = \sum_{r \in V_E} \frac{EDC_r(t)}{EDC(t)} w^r(t). \tag{20}$$

Algorithm 1 presents the pseudo-code of the entire process of FL using our protocol.

Algorithm 1. the HybridFL protocol

Input: maximum number of rounds t_{max}, local epochs per round τ, desired proportion C, response time limit T_{lim}
Output: finalized global model w
// Cloud process: running on the central server
Initializes global model $w(0)$
$quota \leftarrow C \cdot n$
for round $t \leftarrow 1$ to t_{max} **do**
  Distributes $w(t-1)$ to all the edge nodes
  for each edge node r in V_E **in parallel do**
    Computes $C_r(t)$ according to (16)
    edgeUpdate$(r, C_r(t), \tau)$
  end
  Keeps monitoring update count by edge nodes
  if $|S(t)| \geq quota$ or T_{lim} is reached **then**
    // triggers regional aggregation
    Sends aggregation signal to all edge nodes
    edgeAggregation(r)
  end
  // cloud aggregation
  Computes $w(t)$ according to (20)
end
return $w(t)$
// Edge process: running on edge node r
edgeUpdate(r, C_r, τ)
  $q \leftarrow C_r \cdot n_r$
  $Q \leftarrow q$ randomly selected clients in region r
  for each client k in Q **in parallel do**
    clientUpdate(k, r, τ)
    keeps *reporting update count to the cloud*
  end
return
edgeAggregation(r)
  Computes $w^r(t)$ according to (17):
return
// Client process: running on client k
clientUpdate(k, r, τ):
  for epoch $e \leftarrow 1$ to τ **do**
    Updates $w_k^r(t)$ using Gradient Descent method
  end
return

3.3 Convergence Analysis

The convergence of the global model by federated learning has been proved for both the two-layer architecture [12] and the three-layer edge computing systems [13]. However, since we have made modification to the aggregation rules, it is necessary to provide the convergence analysis with the focus on showing the difference compared to the proof provided in the existing work.

Since the regional aggregation in HybridFL is followed instantly by the cloud (global) aggregation, we can mathematically re-formulate the global model $w(t)$ by combining (17) and (20), which yields (21):

$$\begin{aligned} w(t) &= \sum_{r \in V_E} \frac{EDC_r(t)}{EDC(t)} \sum_{k \in V_C^r} \frac{|D_k^r|}{|D^r|} w_k^r(t) \\ &= \sum_{k \in V_C} \frac{EDC_{r(k)}(t)}{EDC(t)} \frac{|D_k^{r(k)}|}{|D^{r(k)}|} w_k^{r(k)}(t) \\ &\triangleq \sum_{k \in V_C} \gamma(k, r(k), t) \cdot w_k^{r(k)}(t). \end{aligned} \tag{21}$$

where $r(k)$ stands for the corresponding edge node connected to client k and the symbol $\gamma(k, r(k), t)$ represents the weight of client k's model during the aggregation. Without ambiguity, in the rest of this paper, we use the abbreviation $\gamma(k, r, t)$ to denote $\gamma(k, r(k), t)$ for brevity.

In (21), the first equality represents the two-level aggregation, namely, the second "$\sum$" represents the edge-level aggregation while the first "$\sum$" represents the cloud-level aggregation. The second equality in (21) transforms the two "$\sum$" into one. This suggests that the entire aggregation in our three-layer MEC system is equivalent to the two-layer FL process as shown in [10] with the only difference lying in weights.

Formally, the HybridFL process is to solve the following global optimization problem at any given round t:

$$\arg\min_w F(w, t) = \sum_{k \in V_C} \gamma(k, r, t) F_k(w), \tag{22}$$

where w denotes the parameters (e.g., weights for neural nets) of the global model to be optimized, $r = r(k)$ is the region index for client k, and $F_k(w)$ is the average loss from the data partition on client k. $F_k(w)$ is calculated by:

$$F_k(w) = \frac{1}{|D_k^{r(k)}|} \sum_{(x_i, y_i) \in D_k^{r(k)}} f(w; x_i, y_i), \tag{23}$$

where $f(\cdot)$ is the loss function and $D_k^{r(k)}$ is the data possessed by client k with $r(k)$ being its region.

We analyze the convergence of our protocol by quantifying the upper bound of $F(w(t), t) - F(w^*, t)$, where w^* denotes the optimal model parameters for our target (see (22)). Due to the space limit, we base our proof on the analysis provided by Wang *et al.* [12] and extend their *Theorems 1 and 2* for the case of our protocol, which yields the Theorem 1* and Theorem 2* in our paper, respectively. We first make the following assumption to facilitate the analysis:

Assumption 1 (Loss function). *$F_k(w)$ is convex, ρ-Lipschitz and β-smooth.*

For the loss functions that do not satisfy the assumption above, Wang *et al.* [12] still validated the effectiveness of FL in such cases. With the assumption, we have: *$F(w, t)$ is convex, ρ-Lipschitz and β-smooth* with regards to w, which can be proved using the triangle inequality based on (22). We also define δ_k as the upper bound of the divergence between the gradients of $F_k(w)$ and $F(w, t)$, and $\bar{\delta}$ as the upper limit of $\delta_k, \forall k \in V_C$:

$$\|\nabla F_k(w) - \nabla F(w, t)\| \leq \delta_k \leq \bar{\delta}. \tag{24}$$

Let z denote the index of epoch (i.e., $z = 1, 2, \ldots, t_{max} \cdot \tau$) ($\tau$ is the number of local epochs in a round). To facilitate the

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

analysis, by $w([z])$ we denote a hypothetical global model as the result of aggregating all $w_k^r([z])$ at epoch z. It is not to be confused with $w(t)$ because $w(t)$ is only visible after the aggregation at the end of a round. Besides, we also consider an auxiliary model $v_t([z])$ learned using *centralized* gradient descent initialized as $w(t-1)$ in the context of round t for optimizing the same target $F(w,t)$. Given z as an epoch in round t (i.e., $z \in ((t-1)\tau, t\tau]$), by the definitions we have:

$$w([z]) = \sum_{k \in V_C} \gamma(k,r,t) w_k^r([z]), \tag{25}$$

where $w_k^r([z])$ is updated from $w_k^r([z-1])$:

$$w_k^r([z]) = w_k^r([z-1]) - \eta \nabla F_k(w_k^r([z-1])). \tag{26}$$

For $v_t([z])$ with $z \in ((t-1)\tau, t\tau]$, we have:

$$v_t([z]) = v_t([z-1]) - \eta \nabla F(v_t([z-1]), t). \tag{27}$$

Now we give our theorem 1*:

Theorem 1* (Loss divergence bound). *for any epoch z in round t, we have*

$$F(w([z]),t) - F(v_t([z]),t) \leq \rho \bar{h}(z-(t-1)\tau), \tag{28}$$

where

$$\bar{h}(x) \triangleq \frac{\bar{\delta}}{\beta}((\eta\beta+1)^x - 1) - \eta\bar{\delta}x. \tag{29}$$

Proof. We base our proof of Theorem 1* on [Lemma 2, ref. [12]] (for proof details, see Appendix (A), which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2020.3040867). □

Theorem 1* gives the theoretical difference in loss between the global model $w([z])$ (by aggregating local models) and the baseline $v_t([z])$ (learned on centralized data) during the training process in round t. Note that $F(w([z]),t) - F(v_t([z]),t) \leq \rho\bar{h}(\tau)$ at $z = t\tau$ and $\bar{h}(1) = 0$. This means that $w([z])$ is equivalent to $v_t([z])$ if the aggregation interval $\tau = 1$. Based on Theorem 1* and recalling that $w(t) = w([t \cdot \tau])$, we now present the convergence upper bound of $w(t)$ in Theorem 2*:

Theorem 2* (Convergence upper bound). *After t rounds with τ epochs in each round, the convergence of the global model is guaranteed by:*

$$F(w(t),t) - F(w^*,t) \leq \frac{1}{t\tau\left(\omega\eta(1-\frac{\beta\eta}{2}) - \frac{\rho\bar{h}(\tau)}{\tau\epsilon^2}\right)}, \tag{30}$$

when the conditions below are satisfied:

1) $\eta \leq \frac{1}{\beta}$
2) $\omega\eta(1-\frac{\beta\eta}{2}) - \frac{\rho\bar{h}(\tau)}{\tau\epsilon^2} > 0$
3) $F(v_t([z]),t) - F(w^*,t) \geq \epsilon$, $\forall z \in ((t-1)\tau, t\tau]$
4) $F(w(t),t) - F(w^*,t) \geq \epsilon$

where $\epsilon > 0$, $\omega \triangleq \min_t \frac{1}{\|v_t([(t-1)\tau]) - w^*\|^2}$, and $\bar{h}(\cdot)$ is defined in (29).

Proof. Condition 1 places a limit on the learning rate η whilst condition 2 implies that the gap between $w([z])$ and $v_t([z])$ needs to be small. Conditions 3 and 4 limit the lower bound of the gap to a positive value ϵ because $w(t)$ is an approximation of w^* given that we perform the aggregation after every τ local epochs and $\tau > 1$. Theorem 2* can be proved based on the conclusion of Theorem 1* combined with the [Lemmas 1, 3 and 4, ref. [12]], and the steps of proof are the same as that provided in [12]. □

Theorem 2* implies that the gap between $w(t)$ and w^* (the optimum) in terms of optimizing the target loss function (see (22)) narrows as the FL process proceeds, i.e., t increases.

3.4 Client Heterogeneity

The heterogeneity of end devices is a common property of practical MEC systems. FL in such systems can involve a vast number of heterogeneous end devices, whose discrepancy in capability (e.g., CPU performance, bandwidth) and reliability has the major impact on the overall efficiency of FL. In this paper, we characterize the heterogeneity of clients by mainly considering their compute performance, bandwidth and reliability.

The compute performance of a client determines how efficiently it conducts local training and can be measured by the CPU frequency (in GHz). Given the same training task and the same size of data partition, clients with lower performance require more time for local training. A certain space of memory is required for any on-device training process. For simplicity we assume that clients only participate when they have sufficient memory, and that memory does not impact clients' performance. After on-device training is completed, local models need to be transmitted to the edge nodes. In this step, device bandwidth is the main factor that determines the communication time between the edge and end devices. Due to the heterogeneity of clients, the time needed for model download/upload and local training differs from client to client. The heterogeneity of clients also differentiates them in energy consumption, which is determined jointly by their power consumption in computation and communication and the time needed for local training (computation) and model transmission (communication). The formulation of device performance and energy consumption is detailed in Section 4.

We also assume that clients are discrepant in reliability because they may drop/opt out by a different probability. Practically, the causes of client drop-out can involve many factors including any subjective/objective reasons, and they all vary from situation to situation. The correlation between these factors and how likely a client drops out is very complicated and beyond the scope of this work. In this paper, we consider client drop-out as an independent event. Our protocol is designed to be completely independent on the distribution of clients' drop-out probability (i.e., designed for reliability-agnostic scenarios).

The design of HybridFL mitigates the negative impact of end devices' heterogeneity and unreliability on the efficiency and effectiveness of FL. The introduction of regional slack factors enables our protocol to modulate the number of model submissions for each region based on the desired

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
Experimental Setup for Federated Learning and the MEC System

Setting	Symbol	Task 1	Task 2
dataset	D	Aerofoil	MNIST
# of features	d	5	28x28
model	w	FCN	LeNet-5
dataset size	$\|D\|$	1503	70k
# of clients	n	15	500
# of edge nodes	m	3	10
data distribution[1]	-	$\mathcal{N}(100, 30^2)$	non-IID, 0.75
client performance	s_k	$\mathcal{N}(0.5, 0.1^2)$	$\mathcal{N}(1.0, 0.3^2)$
client bandwidth	bw_k	$\mathcal{N}(0.5, 0.1^2)$	$\mathcal{N}(1.0, 0.3^2)$
signal-noise ratio	SNR	1e2	1e2
drop-out prob.	dr_k	$\mathcal{N}(\mathbb{E}[dr], 0.05^2)$	$\mathcal{N}(\mathbb{E}[dr], 0.05^2)$
region population	n_r	$\mathcal{N}(5.0, 1.5^2)$	$\mathcal{N}(50, 15^2)$
cloud-edge thrput.	BR	1e3	1e3
max # of rounds	t_{max}	600	400
bits per sample	BPS	6*8*8	28*28*1*8
cycles per bit	CPB	300	400
# of local epochs	τ	5	5
loss function	f	MSE Loss	NLL Loss
learning rate	η	1e-4	1e-3

The units of performance, bandwidth and throughput in the table are GHz, MHz, and Mbps, respectively.

proportion controlled by the cloud. The slack factors are determined without any a priori knowledge on any client's drop-out probability. Besides, the quota-triggered aggregation mechanism in HybridFL allows the cloud to end a round once the quota is met, rather than passively awaiting response from every selected client. This effectively accelerates an FL round and makes the protocol less susceptible to device failure.

4 Experimental Evaluation

We evaluated the effectiveness of the proposed HybridFL in terms of model convergence speed, round efficiency and the global model's accuracy. We also evaluated the energy consumption of end devices, which we consider as an important metric in practice.

4.1 Experiment Setup

In the evaluation, we built a simulated MEC system for Federated Learning as a complete software package. The MEC system was established with simulated parties (i.e., the cloud, edge nodes and end devices) that comprise the three-layer architecture. On-device training in the FL process was implemented using the PyTorch framework. Each group of end devices (clients) are managed by and connected to an edge node via wireless channels, which forms a region, whilst the edge nodes and the cloud are connected through high-speed Ethernet. All the clients and their local network are implemented as being unreliable in the simulated MEC system. The drop-out probability of client k is set as dr_k, which follows a Gaussian distribution (see Table 2) with its mean value set to $\mathbb{E}[dr]$. For client k, the relation between its no-abort probability P_k and its drop-out probability is: $P_k = 1 - dr_k$.

We evaluated HybridFL in two machine learning tasks: *Aerofoil* (Task 1) and *MNIST* (Task 2). For the tasks we configured different MEC environments to test the performance of HybridFL with different scale of end devices and edge nodes on different data distribution. The size of data partitions in each end device in Task 1 follows the Gaussian distribution while in Task 2, we set it to be non-IID by assigning the samples of class y_i, by a probability of 0.75, to the clients with indices $k \equiv y_i \pmod{10}$.

We also implemented two existing protocols in recent literature: *FedAvg* [10] and *HierFAVG* [13]. FedAvg is the primitive FL protocol proposed by Google for the two-layer client/server architecture. HierFAVG is a three-layer FL protocol for Edge Computing systems and adopts a similar training architecture as our protocol by introducing the edge layer that performs the edge-level model aggregation before the global aggregation conducted by the cloud. HierFAVG has no adaptive control over the flow of models. Both the edge and the cloud have to await the responses from all the selected clients. We compare the FL training process driven by these protocols and our HybridFL under the same settings. The parameters in our experimental setting are listed in Table 2.

In this paper, a cycle from the stage of model distribution, local training to model aggregation is called a *federated round*. Note that the global aggregation is performed every federated round by FedAvg and our HybridFL, but HierFAVG performs it after several times of edge-level aggregation (i.e., it runs multiple federated rounds before a cloud aggregation). The cloud-level aggregation interval (κ_2 in reference [13]) for HierFAVG is set to 10, which is shown to be an optimal setting in their work.

For fair comparison, we ran all the protocols for the same number of (federated) rounds, denoted by t_{max}, which also means the same number of total epochs because each client runs the same number of local epochs, denoted by τ, before the edge nodes conduct an edge-level aggregation.

The length of a federated round, denoted by T_{round}, can be formulated as:

$$T_{round} = T_{c2e2c} + \min\{T_{lim}, \max_{k \in V_C'}\{T_k^{comm} + T_k^{train}\}\}, \tag{31}$$

where T_{lim} is the preset limit of response time, which we configured as the time required by an extremely straggling client to finish its local training and communication with an average partition size. Given the performance (denoted by s_k) and bandwidth (denoted by bw_k) of the clients follow the normal distribution with the mean and standard deviation being μ and σ, respectively, the performance and bandwidth of such an extremely straggling client is set to be $\mu - 3\sigma$.

Note that V_C' represents the selected fraction of clients for FedAvg and HierFAVG, but for HybridFL we have $V_C' \equiv S(t)$ because of our quota-triggered aggregation. T_{c2e2c} is the cloud-edge communication time, which is calculated by (32). T_k^{comm} and T_k^{train} are the communication time and local training time of client k, which are calculated by (33) and (34), respectively.

$$T_{c2e2c} = 3 \times \frac{msize \cdot m}{BR}, \tag{32}$$

where BR is the bit rate of the cloud-edge connection while for the edge-client wireless network we obtain its effective

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

TABLE 3
Experimental Results With Task 1: Aerofoil Under Different Environmental Settings of $\mathbb{E}[dr]$ and Client Selection Proportion C

		Stop @t_{max}						Stop @Acc=0.70					
		Best Accuracy			Round length (sec)			Rounds needed			Total time (sec)		
C		0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5
$\mathbb{E}[dr]=0.1$	FedAvg	0.727	0.727	0.727	52.42	73.12	80.73	238	95	56	12515.1	5585.2	4149.7
	HierFAVG	0.727	0.726	0.728	51.56	71.90	81.08	250	80	50	14608.6	4765.8	4285.8
	HybridFL	**0.729**	0.727	0.728	**37.80**	**63.80**	**58.15**	**113**	**75**	**49**	**4254.6**	**3341.4**	**3143.0**
$\mathbb{E}[dr]=0.3$	FedAvg	0.728	0.727	0.727	64.21	83.64	87.90	376	125	74	25442.2	10674.6	6588.8
	HierFAVG	0.727	0.728	0.727	66.74	83.24	88.14	340	130	60	22237.5	12025.9	6132.9
	HybridFL	**0.729**	0.728	**0.728**	**38.94**	**64.83**	**69.84**	**141**	**77**	**51**	**7010.7**	**4994.9**	**3711.6**
$\mathbb{E}[dr]=0.6$	FedAvg	0.711	0.727	0.728	83.54	89.78	90.39	598	233	144	50122.4	21141.3	13108.4
	HierFAVG	0.714	0.727	0.728	81.43	89.91	90.44	590	230	140	48922.2	21623.3	13565.6
	HybridFL	**0.727**	**0.728**	0.728	**65.38**	**73.23**	**84.96**	**160**	**66**	**62**	**10584.1**	**4780.4**	**5488.3**

TABLE 4
Experimental Results With Task 2: MNIST Under Different Environmental Settings of $\mathbb{E}[dr]$ and Client Selection Proportion C

		Stop @t_{max}						Stop @Acc=0.90					
		Best Accuracy			Round length (sec)			Rounds needed			Total time (sec)		
C		0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5
$\mathbb{E}[dr]=0.1$	FedAvg	0.936	0.958	0.962	377.28	378.02	378.02	200	66	32	75166.3	25327.3	12474.6
	HierFAVG	0.936	0.958	0.964	377.65	378.26	378.26	150	40	30	60519.8	18911.3	15128.8
	HybridFL	**0.940**	**0.959**	**0.965**	**63.59**	**96.55**	**140.51**	**124**	**41**	**26**	**7856.9**	**3762.9**	**3730.6**
$\mathbb{E}[dr]=0.3$	FedAvg	0.925	0.951	0.962	378.02	378.02	378.02	230	77	37	87322.3	29485.5	14364.7
	HierFAVG	0.926	0.954	0.962	378.10	378.26	378.26	200	60	30	79432.7	26476.5	15128.8
	HybridFL	**0.940**	**0.959**	**0.966**	**109.72**	**135.96**	**113.20**	**123**	**41**	**25**	**15978.4**	**7148.9**	**4867.4**
$\mathbb{E}[dr]=0.6$	FedAvg	0.901	0.933	0.950	378.02	378.02	**378.02**	376	146	65	142513.1	55568.8	24949.2
	HierFAVG	0.905	0.941	0.952	378.10	378.26	378.26	350	100	60	136171.6	41606.9	26476.5
	HybridFL	**0.937**	**0.960**	**0.963**	**37.59**	**126.15**	380.42	**118**	**41**	**31**	**11743.1**	**7334.8**	**12171.8**

bit rate by applying the Shannon theorem to the corresponding bandwidth bw_k. The multiplicator "3" exists because the model upload typically spends twice as much time as the model download, given that uplink bandwidth is typically 50 percent of the total. The size of the model ($msize$) is set to 5 MB and 10 MB for Tasks 1 and 2, respectively. For FedAvg, $T_{c2e2c} \equiv 0$ because it does not involve the edge layer.

$$T_k^{comm} = 3 \times T_k^{download} = 3 \times \frac{msize}{bw_k \cdot \log(1+SNR)} \tag{33}$$

$$T_k^{train} = \frac{|D_k^r| \cdot \tau \cdot BPS \cdot CPB}{s_k}. \tag{34}$$

The numerator in (34) quantifies the total number of CPU cycles needed for training the local partition D_k^r.

Based on (33) and (34) we can further model the energy consumed by end device for local training:

$$E_k = E_k^{comm} + E_k^{train} = P_{trans} \cdot T_k^{comm} + P_{comp}^{base} s_k^3 \cdot T_k^{train}, \tag{35}$$

where P_{trans} is the power consumption of transmitter and $P_{comp}^{base} s_k^3$ represents the power for on-device computation based on the frequency power model [24]. We set P_{trans} and P_{comp}^{base} to 0.5 and 0.7 Watt respectively based on the benchmarking results reported in ref. [25].

4.2 Experimental Results

We ran the FL process in two ways: i) stop the process at a preset maximum round t_{max}, and ii) stop when a preset accuracy is achieved for the global model. In Table 3 and Table 4, we present the results for task 1 and task 2, respectively, in terms of best model accuracy achieved, average round length (obtained when stopping at t_{max}), the number of rounds needed and the total time duration (for achieving the desired model accuracy). We also investigated the model convergence by comparing the accuracy traces (Figs. 4 and 6) for FedAvg, HierFAVG and HybridFL. Figs. 5 and 7 show the average energy consumption by end devices.

*Task 1: Aerofoil*Aerofoil is a numerical regression task. FL is performed to learn a global Fully-Connected Neural Network (FCN) model from a small group of clients that possess private aerofoil self-noise data[1]. Clients hold different partitions of the data without overlapping and cannot share the data with each other. This task simulates an industrial

1. Airfoil Self-Noise Data Set, UCI. https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

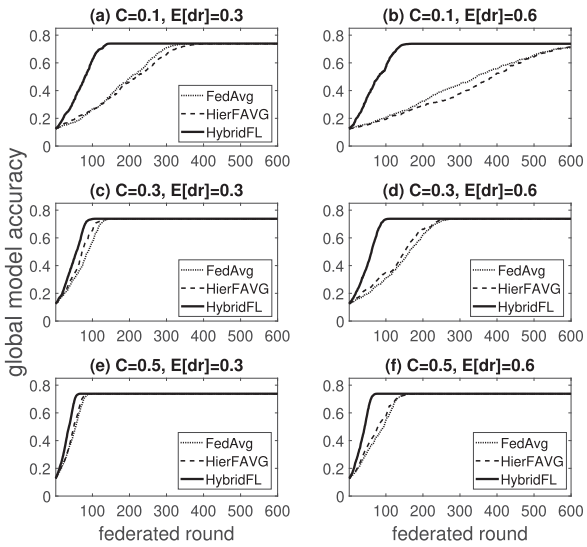


Fig. 4. The trace of model accuracy during the FL process in Task 1: Aerofoil with $C = 0.1, 0.3$ and 0.5. The cloud always keeps the best global model throughout the process. The drop-out probability of end devices follows the Gaussian distribution $\mathcal{N}(\mathbb{E}[dr], 0.05^2)$.

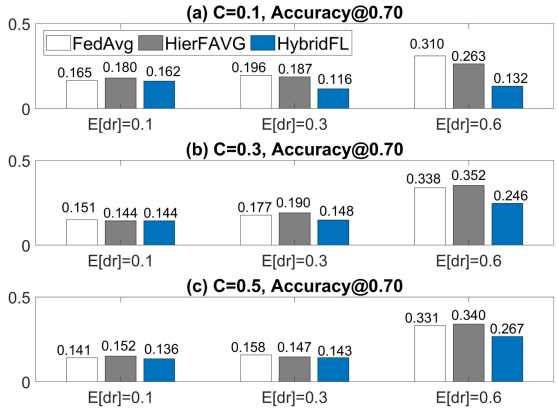


Fig. 5. Comparing the energy consumption (Watt hours) of end devices when running Task 1 (Aerofoil) among FedAvg, HierFAVG, and HybridFL. The values obtained are the average over all the end devices in the MEC system.

scenario where the production data are privacy-sensitive. The size of local partitions follow the Gaussian distribution specified in Table 2.

We ran 600 FL rounds to compare the best accuracy achieved and the average length of a round. The results are shown in the "Stop @t_{max}" column of Table 3. We can see that our protocol effectively shortens the average round length by 6 to 42 percent with slight improvements on the global model's accuracy in most cases. In Fig. 4, we plot the trace of model accuracy over the FL process under the settings of $C \in \{0.1, 0.3, 0.5\}$ and $\mathbb{E}[dr] \in \{0.3, 0.6\}$. From the figure we can observe a solid improvement in model convergence by HybridFL, especially under unstable MEC circumstances where end devices drop out frequently. In the setting of $\mathbb{E}[dr]$=0.6 and the selection proportion C=0.1 (Fig. 4b), the global model can hardly converge in 600 rounds using FedAvg or HierFAVG, but reached its optimum in 200 rounds under the control of our HybridFL protocol.

We also tested the protocols by specifying a target model accuracy as the stop criterion, and observed the number of rounds needed for convergence and the total time duration. The results are shown in the "Stop @Acc" column of Table 3. HybridFL requires much fewer rounds and less time to achieve the accuracy target in Task 1, which yields up to 4× speed-up compared to FedAvg and HierFAVG. In the setting where the clients are mostly unreliable (i.e., $\mathbb{E}[dr]$=0.6), HybridFL can still achieve very fast convergence, requiring only about 1/3 of the rounds needed by HierFAVG. Another benefit of fast convergence is energy conservation. Fig. 5 shows the energy consumption of end devices. We can see that our protocol is most energy consumption friendly to end devices. HybridFL reduces the average energy usage of end devices by roughly 50 percent for Task 1 in the case of $\mathbb{E}[dr] = 0.6$ and $C = 0.1$.

Task 2: MNIST In this task we aim to simulate a scenario in which the image samples are distributed over a relatively large fleet of end devices and they are not shared among the devices or allowed to be uploaded to the servers. This is a realistic scenario for mobile applications that are restricted by the privacy terms. In this experiment, 500 clients and 10 edge nodes are set up for running this task. Besides, to emulate the discrepancy in device users' behaviour (which leads to the biases in the data distribution over devices), we assigned the samples to clients by matching data labels with clients' indices – sample (x_i, y_i) has a 75 percent chance to reside on (one of) the clients whose IDs are congruent to y_i modulo 10 (the MNIST data set has 10 classes). This way the data distribution on each device is far from being IID.

We use the classic convolutional neural net LeNet-5 (consisting of two convolutional layers with max pooling and three fully connected layers) as the model for this image classification task. Again we ran FL for a fixed number of rounds first to observe the best accuracy and round length. The results are shown in the "Stop @t_{max}" column in Table 4, from which we can see that HybridFL outperformed FedAvg and HierFAVG in terms of the accuracy of the global model in all cases, especially when the participating devices are generally unreliable ($\mathbb{E}[dr] = 0.6$). Fig. 6 tracks the accuracy of the global model in the training process under the settings of C= 0.1, 0.3 and 0.5 and $\mathbb{E}[dr]$= 0.3 and 0.6. It can be observed that the convergence of the global model is improved when using HybridFL as the controlling protocol. These results suggest that compared with FedAvg and HierFAVG, HybridFL can achieve the best global model in the fewest number of federated rounds.

We also compare the performance of HybridFL and the baseline protocols by specifying acc= 0.9 as the convergence target for the global model. The results are listed in the right part of Table 4. We can observe from the table that HybridFL significantly reduces the number of rounds and total time needed to achieve the accuracy target, compared with other two protocols. For example, HybridFL achieves a roughly 12× speed-up in the case where $\mathbb{E}[dr]$=0.6 and C=0.1, which represents a situation where the clients may drop out frequently and the participating fraction is restricted.

Some interesting results were observed in the experiments. In both tasks 1 and 2 with $\mathbb{E}[dr] = 0.6$, our protocol requires fewer rounds to converge given $C = 0.5$ than that with $C = 0.3$, but the total time consumption for $C = 0.5$ is longer (see Tables 3 and 4). This is because the extremely

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

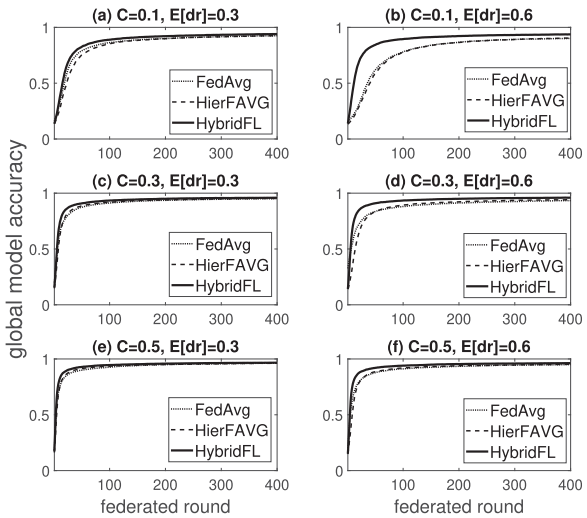


Fig. 6. The trace of accuracy during the FL process in Task 2: MNIST with $C = 0.1, 0.3$ and 0.5. The cloud always keeps the best global model throughout the process. The drop-out probability of end devices follows the Gaussian distribution $\mathcal{N}(\mathbb{E}[dr], 0.05^2)$.

high drop-out probability (0.6 on average in the cases) of clients makes it almost impossible to engage 50 percent of the them (given $C = 0.5$) in training, even with the modulation of the regional slack factors. This is the case $|S(t)| < C \cdot n$. In such a case, the edge nodes and the cloud have to wait until the preset round-time limit is reached, and thus the round length is prolonged. To some extent, this observation explains why it is suggested in literature [23] that the selection proportion C should not be set too large.

Device energy usage can be a key factor that affects the willingness of device owners to participate in the FL training. Fig. 7 shows the average energy consumption of an end device as a participant in the FL process to achieve the preset accuracy target 0.9 for Task 2. We find that the advantage of HybridFL in energy saving in Task 2 is not as prominent as in Task 1. Yet our protocol still managed to retain the on-device energy usage at the lowest level. This is because it enables much faster convergence (therefore less total training time for devices) than the baseline protocols. In practice, the energy-saving feature of HybridFL can help attract more end devices in each round.

The evaluation of the proposed protocol (HybridFL) with the two machine learning tasks under different environment

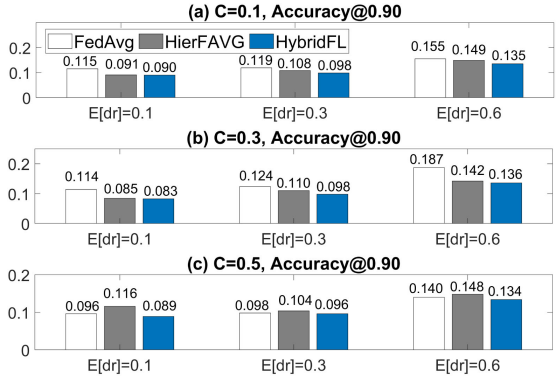


Fig. 7. Comparing the average on-device energy (Watt hours) consumed for Task 2: MNIST in local training when using FedAvg, HierFAVG and HybridFL as the control protocols. The values obtained are the average over all the end devices in the MEC system.

settings demonstrates its effectiveness in terms of boosting the efficiency of FL, improving the global model's quality and saving on-device energy consumption in a three-layer MEC system. The reasons behind these improvements are three-fold. First, the quota-triggered regional aggregation in HybridFL effectively prevents the situation where some regions with extremely unreliable clients slow down the entire FL process. Second, we enable each edge node to modulate its regional quota based on its slack factor to improve the robustness of FL against client drop-out. Third, the cloud (i.e., global) aggregation is designed to be performed immediately after regional aggregation so that the global exchange of the model is made as early as possible.

5 Conclusion

Thanks to the ever-increasing capacity of compute, storage and bandwidth at the edge of network, it has been a prominent trend that more and more end devices are infiltrated by the power of artificial intelligence. Meanwhile, the rising concerns about data privacy are changing the way we develop machine learning techniques and also reveal the great potential of using Federated Learning as a promising privacy-preserving solution. In this paper, we adapt FL to the mobile edge computing systems, aiming to improve both effectiveness and efficiency. We design a three-layer FL protocol called HybridFL to enable two levels of model aggregation to boost efficiency and mitigate the impact by the unreliable nature of end devices through modulating client selection in a region-wise manner, which results in a reasonable number of local updates as desired by the cloud. We conducted extensive experiments and the results demonstrate that HybridFL significantly improves FL in the MEC system by shortening the average length of a round, speeding up the convergence of the global model, promoting the model accuracy, and reducing device-side energy consumption.

In the future, we plan to extend our work to more complex system architectures which have different hierarchies and to diverse FL participants which have different roles. As another part of our future work, we plan to investigate how to improve the effectiveness of local training on each device without breaching the privacy constraints.

Acknowledgments

This work was supported in part by the Worldwide Byte Security Information Technology Company Ltd., Guangdong Project (Grant No. 2018B030325002), Key-Area Research and Development Program of Guangdong Province (Grant No. 2020B010164003), Guangzhou Science and Technology Program Key Project (Grant Nos. 202007040002 and 2019020 10040), and Guangzhou Development Zone Science and Technology (Grant No. 2018GH17).

References

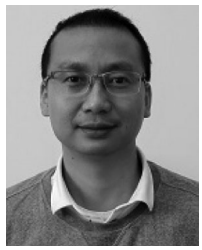
[1] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. Workshop Mobile Edge Commun.*, 2018, pp. 31–36.

[2] Q. Meng, K. Wang, X. He, and M. Guo, "QoE-driven big data management in pervasive edge computing environment," *Big Data Mining Anal.*, vol. 1, no. 3, pp. 222–233, 2018.

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.

[3] C. Pemberton, "3 AI Trends for Enterprise Computing," 2017. [Online] Available: https://www.gartner.com/smarterwithgartner/3-ai-trends-for-enterprise-computing/
[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.
[5] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, Thirdquarter 2017.
[6] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020.
[7] Y. Yu, M. Li, L. Liu, Y. Li, and J. Wang, "Clinical big data and deep learning: Applications, challenges, and future outlooks," *Big Data Mining Anal.*, vol. 2, no. 4, pp. 288–305, 2019.
[8] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, and Q. V. Le, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*. 2012, pp. 1223–1231.
[9] B. Recht, C. Re, S. Wright, and F. Niu, "HOGWILD: A lock-free approach to parallelizing stochastic gradient descent," in *Proc. Int Conf. Neural Inf. Process. Syst.*, 2011, pp. 693–701.
[10] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Arti. Intell. Statist.*, 2017, pp. 1273–1282.
[11] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proc. 29th Conf. Neural Inf. Process. Syst.*, 2016, pp. 1–10.
[12] 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.
[13] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.
[14] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. 19th Int. Conf. Comput. Statist.*, 2010, pp. 177–186.
[15] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Rev.*, vol. 60, no. 2, pp. 223–311, 2018.
[16] J. Wu, W. Huang, J. Huang, and T. Zhang, "Error compensated quantized SGD and its applications to large-scale distributed optimization," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5321–5329.
[17] S. Zheng *et al.*, "Asynchronous stochastic gradient descent with delay compensation," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, pp. 4120–4129, Aug. 2017.
[18] V. Smith, C. K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4424–4434.
[19] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019, *arXiv:1903.03934*.
[20] M. R. Sprague *et al.*, "Asynchronous federated learning for geospatial applications," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2018, pp. 21–28.
[21] W. Wu, L. He, W. Lin, and S. Jarvis, "SAFA: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Comput.*, to be published, doi: 10.1109/TC.2020.2994391
[22] K. Bonawitz *et al.* "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*.
[23] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.
[24] W. Lin, W. Wu, H. Wang, J. Z. Wang, and C. H. Hsu, "Experimental and quantitative analysis of server power model for cloud data centers," *Future Gener. Comput. Syst.*, vol. 86, pp. 940–950, 2018.
[25] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, 2010, pp. 1–14.

Wentai Wu (Student Member, IEEE) received the bachelor's and master's degrees in computer science from the South China University of Technology, in 2015 and 2018, respectively. He is currently working toward the PhD degree with the Department of Computer Science, University of Warwick, United Kingdom, supervised by Dr. Ligang He. His research interests mainly include parallel and distributed computing, distributed machine learning, and energy-efficient computing.



Ligang He (Member, IEEE) received the PhD degree in computer science from the University of Warwick, United Kingdom. He worked as a post-doctoral researcher at the University of Cambridge, United Kingdom. He then joined the Department of Computer Science at the University of Warwick as an assistant professor, and was then promoted to associate professor, where currently a reader with the department. He has published more than 130 articles in journals and conferences, such as *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *ACM Transactions on Architecture and Code Optimization*, IPDPS, ICPP, SC, and VLDB. His research interests focus on parallel and distributed processing, and big data processing.

Weiwei Lin received the BS and MS degrees from Nanchang University, in 2001 and 2004, respectively, and the PhD degree in computer application from South China University of Technology, in 2007. Currently, he is a professor with the School of Computer Science and Engineering, South China University of Technology. His research interests include distributed systems, cloud computing, big data computing, and AI application technologies. He has published more than 80 papers in refereed journals and conference proceedings.

Rui Mao received the BS and MS degrees in computer science from the University of Science and Technology of China, China, in 1997, and 2000, respectively, the MS degree in statistics, in 2006, and the PhD degree in computer science both from the University of Texas at Austin, USA, in 2007. After three years of working at the Oracle USA Corporation, he joined Shenzhen University (SZU), China, in 2010. He is currently a professor and the vice dean at the College of Computer Science and Software Engineering, SZU. His research interest includes universal data management and analysis in metric space, and high performance computing.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on July 14,2021 at 12:11:13 UTC from IEEE Xplore. Restrictions apply.