Federated Learning Over Wireless Networks: Convergence Analysis and Resource Allocation

Canh T. Dinh, Nguyen H. Tran, *Senior Member, IEEE*, Minh N. H. Nguyen, *Member, IEEE*, Choong Seon Hong, *Senior Member, IEEE*, Wei Bao, *Member, IEEE*, Albert Y. Zomaya, *Fellow, IEEE*, and Vincent Gramoli, *Member, IEEE*

***Abstract*— There is an increasing interest in a fast-growing machine learning technique called Federated Learning (FL), in which the model training is distributed over mobile user equipment (UEs), exploiting UEs' local computation and training data. Despite its advantages such as preserving data privacy, FL still has challenges of heterogeneity across UEs' data and physical resources. To address these challenges, we first propose FEDL, a FL algorithm which can handle heterogeneous UE data without further assumptions except strongly convex and smooth loss functions. We provide a convergence rate characterizing the trade-off between local computation rounds of each UE to update its local model and global communication rounds to update the FL global model. We then employ FEDL in wireless networks as a resource allocation optimization problem that captures the trade-off between FEDL convergence wall clock time and energy consumption of UEs with heterogeneous computing and power resources. Even though the wireless resource allocation problem of FEDL is non-convex, we exploit this problem's structure to decompose it into three sub-problems and analyze their closed-form solutions as well as insights into problem design. Finally, we empirically evaluate the convergence of FEDL with PyTorch experiments, and provide extensive numerical results for the wireless resource allocation sub-problems. Experimental results show that FEDL outperforms the vanilla FedAvg algorithm in terms of convergence rate and test accuracy in various settings.**

Manuscript received March 10, 2020; revised July 20, 2020 and September 10, 2020; accepted October 20, 2020; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor V. Aggarwal. This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.02-2019.321. This article was presented in part at the IEEE INFOCOM 2019. Date of publication November 17, 2020; date of current version February 17, 2021. *(Corresponding authors: Nguyen H. Tran; Choong Seon Hong.)*

Canh T. Dinh, Nguyen H. Tran, Wei Bao, and Albert Y. Zomaya are with the School of Computer Science, The University of Sydney, Sydney, NSW 2006, Australia (e-mail: tdin6081@uni.sydney.edu.au; nguyen.tran@sydney.edu.au; wei.bao@sydney.edu.au; albert.zomaya@sydney.edu.au).

Minh N. H. Nguyen is with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 02447, South Korea, and also with the Vietnam - Korea University of Information and Communication Technology, The University of Da Nang, Da Nang 550000, Vietnam (e-mail: minhnhn@khu.ac.kr).

Choong Seon Hong is with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 130-701, South Korea (e-mail: cshong@khu.ac.kr).

Vincent Gramoli is with the School of Computer Science, The University of Sydney, Sydney, NSW 2006, Australia, and also with the Distributed Computing Lab at EPFL, CH-1015 Lausanne, Switzerland (e-mail: vincent.gramoli@epfl.ch).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TNET.2020.3035770

***Index Terms*— Distributed machine learning, federated learning, optimization decomposition.**

I. INTRODUCTION

THE significant increase in the number of cutting-edge mobile and Internet of Things (IoT) devices results in the phenomenal growth of the data volume generated at the edge network. It has been predicted that in 2025 there will be 80 billion devices connected to the Internet and the global data will achieve 180 trillion gigabytes [2]. However, most of this data is privacy-sensitive in nature. It is not only risky to store this data in data centers but also costly in terms of communication. For example, location-based services such as the app Waze [3], can help users avoid heavy-traffic roads and thus reduce congestion. However, to use this application, users have to share their own locations with the server and it cannot guarantee that the location of drivers is kept safely. Besides, in order to suggest the optimal route for drivers, Waze collects a large number of data including every road driven to transfer to the data center. Transferring this amount of data requires a high expense in communication and drivers' devices to be connected to the Internet continuously.

In order to maintain the privacy of consumer data and reduce the communication cost, it is necessary to have an emergence of a new class of machine learning techniques that shifts computation to the edge network where the privacy of data is maintained. One such popular technique is called Federated Learning (FL) [4]. This technology allows users to collaboratively build a shared learning model while preserving all training data on their user equipment (UE). In particular, a UE computes the updates to the current global model on its local training data, which is then aggregated and fed-back by a central server, so that all UEs have access to the same global model to compute their new updates. This process is repeated until an accuracy level of the learning model is reached. In this way, the user data privacy is well protected because local training data are not shared, which thus differs FL from conventional approaches in data acquisition, storage, and training.

There are several reasons why FL is attracting plenty of interests. Firstly, modern smart UEs are now able to handle heavy computing tasks of intelligent applications as they

are armed with high-performance central processing units (CPUs), graphics processing units (GPUs) and integrated AI chips called neural processing units (e.g., Snapdragon 845, Kirin 980 CPU and Apple A12 Bionic CPU [5]). Being equipped with the latest computing resources at the edge, the model training can be updated locally leading to the reduction in the time to upload raw data to the data center. Secondly, the increase in storage capacity, as well as the plethora of sensors (e.g., cameras, microphones, GPS) in UEs enables them to collect a wealth amount of data and store it locally. This facilitates unprecedented large-scale flexible data collection and model training. With recent advances in edge computing, FL can be more easily implemented in reality. For example, a crowd of smart devices can proactively sense and collect data during the day hours, then jointly feedback and update the global model during the night hours, to improve the efficiency and accuracy for next-day usage. We envision that such this approach will boost a new generation of smart services, such as smart transportation, smart shopping, and smart hospital.

Despite its promising benefits, FL comes with new challenges to tackle. On one hand, the number of UEs in FL can be large and the data generated by UEs have diverse distributions [4]. Designing efficient algorithms to handle statistical heterogeneity with convergence guarantee is thus a priority question. Recently, several studies [4], [6], [7] have used de facto optimization algorithms such as Gradient Descent (GD), Stochastic Gradient Descent (SGD) to enable devices' local updates in FL. One of the most well-known methods named FedAvg [4] which uses average SGD updates was experimentally shown to perform well in heterogeneous UE data settings. However, this work lacks theoretical convergence analysis. By leveraging edge computing to enable FL, [7] proposed algorithms for heterogeneous FL networks by using GD with bounded gradient divergence assumption to facilitate the convergence analysis. In another direction, the idea of allowing UEs to solve local problems in FL with arbitrary optimization algorithm to obtain a local accuracy (or inexactness level) has attracted a number of researchers [8], [9]. While [8] uses primal-dual analysis to prove the algorithm convergence under any distribution of data, the authors of [9] propose adding proximal terms to local functions and use primal analysis for convergence proof with a local dissimilarity assumption, a similar idea of bounding the gradient divergence between local and global loss functions.

While all of the above FL algorithms' complexities are measured in terms of the number of local and global update rounds (or iterations), the wall clock time of FL when deployed in a wireless environment mainly depends on the number of UEs and their diverse characteristics, since UEs may have different hardware, energy budget, and wireless connection status. Specifically, the total wall-clock training time of FL includes not only the UE computation time (which depend on UEs' CPU types and local data sizes) but also the communication time of all UEs (which depends on UEs' channel gains, transmission power, and local data sizes). Thus, to minimize the wall-clock training time of FL, a careful resource allocation problem for FL over wireless networks needs to consider not only the FL parameters such as accuracy level for computation-communication trade-off, but also allocating the UEs' resources such as power and CPU cycles with respect to wireless conditions. From the motivations above, our contributions are summarized as follows:

- We propose a new FL algorithm with only assumption of strongly convex and smooth loss functions, named FEDL. The crux of FEDL is a new local surrogate function, which is designed for each UE to solve its local problem approximately up to a local accuracy level θ, and is characterized by a hyper-learning rate η. Using primal convergence analysis, we show the linear convergence rate of FEDL by controlling η and θ, which also provides the trade-off between the number of local computation and global communication rounds. We then employ FEDL, using both strongly convex and non-convex loss functions, on PyTorch to verify its performance with several federated datasets. The experimental results show that FEDL outperforms the vanilla FedAvg [4] in terms of training loss, convergence rate and test accuracy.
- We propose a resource allocation problem for FEDL over wireless networks to capture the trade-off between the wall clock training time of FEDL and UE energy consumption by using the Pareto efficiency model. To handle the non-convexity of this problem, we exploit its special structure to decompose it into three sub-problems. The first two sub-problems relate to UE resource allocation over wireless networks, which are transformed to be convex and solved separately; then their solutions are used to obtain the solution to the third sub-problem, which gives the optimal η and θ of FEDL. We derive their closed-form solutions, and characterize the impact of the Pareto-efficient controlling knob to the optimal: (i) computation and communication training time, (ii) UE resource allocation, and (iii) hyper-learning rate and local accuracy. We also provide extensive numerical results to examine the impact of UE heterogeneity and Pareto curve of UE energy cost and wall clock training time.

The rest of this paper is organized as follows. Section II discusses related works. Section III contains system model. Sections IV and V provide the proposed FL algorithm's analysis and resource allocation over wireless networks, respectively. Experimental performance of FEDL and numerical results of the resource allocation problem are provided in Section VI and Section VII, respectively. Section VIII concludes our work. All theoretical proofs can be found in the supplementary material.

II. Related Works

Due to Big Data applications and complex models such as Deep Learning, training machine learning models needs to be distributed over multiple machines, giving rise to researches on decentralized machine learning [10]–[14]. However, most of the algorithms in these works are designed for machines having balanced and/or independent and identically distributed (i.i.d.) data. Realizing the lack of studies in dealing with

unbalanced and heterogeneous data distribution, an increasing number of researchers place interest in studying FL, a state-of-the-art distributed machine learning technique [4], [7], [9], [15]–[18]. This technique takes advantage of the involvement of a large number of devices where data are generated locally, which makes them statistically heterogeneous in nature. As a result, designing algorithms with global model's convergence guarantee becomes challenging. There are two main approaches to overcome this problem.

The first approach is based on de facto algorithm SGD with a fixed number of local iterations on each device [4]. Despite its feasibility, these studies still have limitations as lacking the convergence analysis. The work in [7], on the other hand, used GD and additional assumptions on Lipschitz local functions and bounded gradient divergence to prove the algorithm convergence.

Another useful approach to tackling the heterogeneity challenge is to allow UEs to solve their primal problems approximately up to a local accuracy threshold [9], [16], [17]. Their works show that the main benefit of this approximation approach is that it allows flexibility in the compromise between the number of rounds run on the local model update and the communication to the server for the global model update. [17] exploits proximal stochastic variance reduced gradient methods for both convex and non-convex FL. While the authors of [9] use primal convergence analysis with bounded gradient divergence assumption and show that their algorithm can apply to non-convex FL setting, [16] uses primal-dual convergence analysis, which is only applicable to FL with convex problems.

From a different perspective, many researchers have recently focused on the efficient communications between UEs and edge servers in FL-supported networks [1], [7], [19]–[24]. The work [7] proposes algorithms for FL in the context of edge networks with resource constraints. While there are several works [25], [26] that study minimizing communicated messages for each global iteration update by applying sparsification and quantization, it is still a challenge to utilize them in FL networks. For example, [19] uses the gradient quantization, gradient sparsification, and error accumulation to compress gradient message under the wireless multiple-access channel with the assumption of noiseless communication. The work [20] studies a similar quantization technique to explore convergence guarantee with low-precision training. [24] considers joint learning with a subset of users and wireless factors such as packet errors and the availability of wireless resources while [21] focuses on using cell-free massive MIMO to support FL. [22], [23] apply a Stackelberg game to motivate the participation of the clients during the aggregation. Contrary to most of these works which make use of existing, standard FL algorithms, our work proposes a new one. Nevertheless, these works lack studies on unbalanced and heterogeneous data among UEs. We study how the computation and communication characteristics of UEs can affect their energy consumption, training time, convergence and accuracy level of FL, considering heterogeneous UEs in terms of data size, channel gain and computational and transmission power capabilities.

III. System Model

We consider a wireless multi-user system which consists of one edge server and a set $\mathcal{N}$ of N UEs. Each participating UE n stores a local dataset $\mathcal{D}_n$, with its size denoted by D_n. Then, we can define the total data size by $D = \sum_{n=1}^{N} D_n$. In an example of the supervised learning setting, at UE n, $\mathcal{D}_n$ defines the collection of data samples given as a set of input-output pairs $\{x_i, y_i\}_{i=1}^{D_n}$, where $x_i \in \mathbb{R}^d$ is an input sample vector with d features, and $y_i \in \mathbb{R}$ is the labeled output value for the sample x_i. The data can be generated through the usage of UE, for example, via interactions with mobile apps.

In a typical learning problem, for a sample data $\{x_i, y_i\}$ with input x_i (e.g., the response time of various apps inside the UE), the task is to find the *model parameter* $w \in \mathbb{R}^d$ that characterizes the output y_i (e.g., label of edge server load, such as high or low, in next hours) with the loss function $f_i(w)$. The loss function on the data set of UE n is defined as

$$F_n(w) \ := \frac{1}{D_n} \sum\nolimits_{i \in \mathcal{D}_n} f_i(w).$$

Then, the learning model is the minimizer of the following global loss function minimization problem

$$\min_{w \in \mathbb{R}^d} F(w) := \sum\nolimits_{n=1}^{N} p_n F_n(w), \tag{1}$$

where $p_n := \frac{D_n}{D}, \forall n$.

Assumption 1: $F_n(\cdot)$ is L-smooth and β-strongly convex, $\forall n$, respectively, as follows, $\forall w, w' \in \mathbb{R}^d$:

$$F_n(w) \leq F_n(w') + \left\langle \nabla F_n(w'), w - w' \right\rangle + \frac{L}{2} \left\| w - w' \right\|^2$$
$$F_n(w) \geq F_n(w') + \left\langle \nabla F_n(w'), w - w' \right\rangle + \frac{\beta}{2} \left\| w - w' \right\|^2.$$

Throughout this paper, $\langle w, w' \rangle$ denotes the inner product of vectors w and w' and $\|\cdot\|$ is Euclidean norm. We note that strong convexity and smoothness in Assumption 1, also used in [7], can be found in a wide range of applications such as l_2-regularized linear regression model with $f_i(w) = \frac{1}{2}(\langle x_i, w \rangle - y_i)^2 + \frac{\beta}{2} \|w\|^2, y_i \in \mathbb{R}$, and l_2-regularized logistic regression with $f_i(w) = \log\big(1 + \exp(-y_i \langle x_i, w \rangle)\big) + \frac{\beta}{2} \|w\|^2, y_i \in \{-1, 1\}$. We also denote $\rho := \frac{L}{\beta}$ the condition number of $F_n(\cdot)$'s Hessian matrix.

IV. Federated Learning Algorithm Design

In this section, we propose a FL algorithm, named FEDL, as presented in Algorithm 1. To solve problem (1), FEDL uses an iterative approach that requires K_g *global rounds* for global model updates. In each global round, there are interactions between the UEs and edge server as follows.

UEs update local models: In order to obtain the local model w_n^t at a global round t, each UE n first receives the feedback information w^{t-1} and $\nabla \bar{F}^{t-1}$ (which will be defined later in (4) and (5), respectively) from the server, and then minimize its following surrogate function (line 3)

$$\min_{w \in \mathbb{R}^d} J_n^t(w) \ := F_n(w) \ + \left\langle \eta \nabla \bar{F}^{t-1} - \nabla F_n(w^{t-1}), w \right\rangle. \tag{2}$$

Algorithm 1 FEDL

1: **Input:** w^0, $\theta \in [0,1]$, $\eta > 0$.
2: **for** $t = 1$ to K_g **do**
3: **Computation:** Each UE n receives w^{t-1} and $\nabla \bar{F}^{t-1}$ from the server, and solves (2) in K_l rounds to achieve θ-approximation solution w_n^t satisfying (3).
4: **Communication:** UE n transmit w_n^t and $\nabla F_n(w_n^t)$, $\forall n$, to the edge server.
5: **Aggregation and Feedbacks:** The edge server updates the global model w^t and $\nabla \bar{F}^t$ as in (4) and (5), respectively, and then fed-backs them to all UEs.

One of the key ideas of FEDL is UEs can solve (2) approximately to obtain an approximation solution w_n^t satisfying

$$\left\|\nabla J_n^t(w_n^t)\right\| \leq \theta \left\|\nabla J_n^t(w^{t-1})\right\|, \forall n, \tag{3}$$

which is parametrized by a local accuracy $\theta \in (0,1)$ that is common to all UEs. This local accuracy concept resembles the approximate factors in [8], [27]. Here $\theta = 0$ means the local problem (2) is required to be solved optimally, and $\theta = 1$ means no progress for local problem, e.g., by setting $w_n^t = w^{t-1}$. The surrogate function $J_n^t(.)$ (2) is motivated from the scheme Distributed Approximate NEwton (DANE) proposed in [11]. However, DANE requires (i) the global gradient $\nabla F(w^{t-1})$ (which is not available at UEs or server in FL context), (ii) additional proximal terms (i.e., $\frac{\mu}{2}\left\|w - w^{t-1}\right\|^2$), and (iii) solving local problem (2) exactly (i.e., $\theta = 0$). On the other hand, FEDL uses (i) the *global gradient estimate* $\nabla \bar{F}^{t-1}$, which can be measured by the server from UE's information, instead of exact but unrealistic $\nabla F(w^{t-1})$, (ii) avoids using proximal terms to limit additional controlling parameter (i.e., μ), and (iii) flexibly solves local problem approximately by controlling θ. Furthermore, we have $\nabla J_n^t(w) = \nabla F_n(w) + \eta \nabla \bar{F}^{t-1} - \nabla F_n(w^{t-1})$, which includes both local and global gradient estimate weighted by a controllable parameter η. We will see later how η affects to the convergence of FEDL. Compared to the vanilla FedAvg, FEDL requires more information (UEs sending not only w_n^t but also $\nabla F_n(w_n^t)$) to obtain the benefits of a) theoretical linear convergence and b) experimental faster convergence, which will be shown in later sections. And we will also show that with Assumption 1, the theoretical analysis of FEDL does not require the gradient divergence bound assumption, which is typically required in non-strongly convex cases as in [7, Definition 1], [9, Assumption 1].

Edge server updates global model: After receiving the local model w_n^t and gradient $\nabla F_n(w_n^t)$, $\forall n$, the edge server aggregates them as follows

$$w^t := \sum\nolimits_{n=1}^{N} p_n w_n^t, \tag{4}$$

$$\nabla \bar{F}^t := \sum\nolimits_{n=1}^{N} p_n \nabla F_n(w_n^t) \tag{5}$$

and then broadcast w^t and $\nabla \bar{F}^t$ to all UEs (line 5), which are required for participating UEs to minimize their surrogate J_n^{t+1} in the next global round $t+1$. We see that the edge server does not access the local data $\mathcal{D}_n$, $\forall n$, thus preserving data privacy. For an arbitrary small constant $\epsilon > 0$, the problem (1) achieves a global model convergence w^t when its satisfies

$$F(w^t) - F(w^*) \leq \epsilon, \quad \forall t \geq K_g, \tag{6}$$

where w^* is the optimal solution to (1).

Next, we will provide the convergence analysis for FEDL. We see that $J_n^t(w)$ is also β-strongly convex and L-smooth as $F_n(\cdot)$ because they have the same Hessian matrix. With these properties of $J_n^t(w)$, we can use GD to solve (2) as follows

$$z^{k+1} = z^k - h_k \nabla J_n^t(z^k), \tag{7}$$

where z_k is the local model update and h_k is a predefined learning rate at iteration k, which has been shown to generate a convergent sequence $(z_k)_{k \geq 0}$ satisfying a linear convergence rate [28] as follows

$$J_n^t(z_k) - J_n^t(z^*) \leq c(1-\gamma)^k \left(J_n^t(z_0) - J_n^t(z^*)\right), \tag{8}$$

where z^* is the optimal solution to the local problem (2), and c and $\gamma \in (0,1)$ are constants depending on ρ.

Lemma 1: With Assumption 1 and the assumed linear convergence rate (8) *with $z_0 = w^{t-1}$, the number of local rounds K_l for solving* (2) *to achieve a θ-approximation condition* (3) *is*

$$K_l = \frac{2}{\gamma} \log \frac{C}{\theta}, \tag{9}$$

where $C := c\rho$.

Theorem 1: With Assumption 1, the convergence of FEDL is achieved with linear rate

$$F(w^t) - F(w^*) \leq (1-\Theta)^t (F(w^{(0)}) - F(w^*)), \tag{10}$$

when $\Theta \in (0,1)$, and

$$\Theta := \frac{\eta(2(\theta-1)^2 - (\theta+1)\theta(3\eta+2)\rho^2 - (\theta+1)\eta\rho^2)}{2\rho\left((1+\theta)^2\eta^2\rho^2 + 1\right)}. \tag{11}$$

Corollary 1: The number of global rounds for FEDL to achieve the convergence satisfying (6) *is*

$$K_g = \frac{1}{\Theta} \log \frac{F(w^0) - F(w^*)}{\epsilon}. \tag{12}$$

The proof of this corollary can be shown similarly to that of Lemma 1. We have some following remarks:

1) The convergence of FEDL can always be obtained by setting sufficiently small values of both η and $\theta \in (0,1)$ such that $\Theta \in (0,1)$. While the denominator of (11) is greater than 2, its numerator can be rewritten as $2\eta(A-B)$, where $A = 2(\theta-1)^2 - (\theta+1)\theta(3\eta+2)\rho^2$ and $B = (\theta+1)\eta\rho^2$. Since $\lim_{\theta \to 0} A = 2$ and $\lim_{\theta,\eta \to 0} B = 0$, there exists small values of θ and η such that A-B > 0, thus $\Theta > 0$. On the other hand, we have $\lim_{\eta \to 0} \Theta = 0$; thus, there exists a small value of η such that $\Theta < 1$. We note that $\Theta \in (0,1)$ is only the sufficient condition, but not the necessary condition, for the convergence of FEDL. Thus, there exist possible hyper-parameter settings such that FEDL converges but $\Theta \notin (0,1)$.
2) There is a convergence trade-off between the number of local and global rounds characterized by θ: small θ makes

large K_l, yet small K_g, according to (9) and (12), respectively. This trade-off was also observed by authors of [8], though their technique (i.e., primal-dual optimization) is different from ours.

3) While θ affects to both local and global convergence, η only affects to the global convergence rate of FEDL. If η is small, then Θ is also small, thus inducing large K_g. However, if η is large enough, Θ may not be in $(0, 1)$, which leads to the divergence of FEDL. We call η the *hyper-learning rate* for the global problem (1).
4) The condition number ρ also affects to the FEDL convergence: if ρ is large (i.e., poorly conditioned problem (2)), both η and θ should be sufficiently small in order for $\Theta \in (0, 1)$ (i.e., slow convergence rate.) This observation is well-aligned to traditional optimization convergence analysis [29, Chapter 9].
5) In this work, the theory of FEDL is applicable to (i) full data passing using GD, (ii) the participation of all UEs, and (iii) strongly convex and smooth loss functions. However, using mini-batch is a common practice in machine learning to reduce the computation load at the UEs. On the other hand, choosing a subset of participating UEs in each global iteration is a practical approach to reduce the straggler effect, in which the run-time of each iteration is limited by the "slowest" UEs (the straggler) because heterogeneous UEs compute and communicate at different speeds. Finally, non-convex loss functions capture several essential machine learning tasks using neural networks. In Section VII, we will experimentally show that FEDL works well with (i) mini-batch, (ii) subset of UEs samplings, and (iii) non-convex loss functions.

The time complexity of FEDL is represented by K_g communication rounds and computation complexity is $K_g K_l$ computation rounds. When implementing FEDL over wireless networks, the wall clock time of each communication round can be significantly larger than that of computation if the number of UEs increases, due to multi-user contention for wireless medium. In the next section, we will study the UE resource allocation to enable FEDL over wireless networks.

V. FEDL Over Wireless Networks

In this section, we first present the system model and problem formulation of FEDL over a time-sharing wireless environment. We then decompose this problem into three sub-problems, derive their closed-form solutions, reveal the hindsights, and provide numerical support.

A. System Model

At first, we consider synchronous communication which requires all UEs to finish solving their local problems before entering the communication phase. During the communication phase, the model's updates are transferred to the edge server by using a wireless medium sharing scheme. In the communication phase, each global round consists of computation and communication time which includes uplink and downlink ones. In this work, however, we do not consider the downlink communication time as it is negligible compared to the uplink one. The reason is that the downlink has larger bandwidth than the uplink and the edge server power is much higher than UE's transmission power. Besides, the computation time only depends on the number of local rounds, and thus θ, according to (9). Denoting the time of one local round by T_{cp}, i.e., the time to computing one local round (8), then the computation time in one global round is $K_l\, T_{cp}$. Denoting the communication time in one global round by T_{co}, the wall clock time of one global round of FEDL is defined as

$$T_g := T_{co} + K_l\, T_{cp}.$$

1) Computation Model: We denote the number of CPU cycles for UE n to execute one sample of data by c_n, which can be measured offline [30] and is known a priori. Since all samples $\{x_i, y_i\}_{i \in \mathcal{D}_n}$ have the same size (i.e., number of bits), the number of CPU cycles required for UE n to run one local round is $c_n D_n$. Denote the CPU-cycle frequency of the UE n by f_n. Then the CPU energy consumption of UE n for one local round of computation can be expressed as follows [31]

$$E_{n,cp} = \sum\nolimits_{i=1}^{c_n D_n} \frac{\alpha_n}{2} f_n^2 = \frac{\alpha_n}{2} c_n D_n f_n^2, \tag{13}$$

where $\alpha_n/2$ is the effective capacitance coefficient of UE n's computing chipset. Furthermore, the computation time per local round of the UE n is $\frac{c_n D_n}{f_n}$, $\forall n$. We denote the vector of f_n by $f \in \mathbb{R}^n$.

2) Communication Model: In FEDL, regarding to the communication phase of UEs, we consider a time-sharing multi-access protocol (similar to TDMA) for UEs. We note that this time-sharing model is not restrictive because other schemes, such as OFDMA, can also be applied to FEDL. The achievable transmission rate (nats/s) of UE n is defined as follows:

$$r_n = B \ln\big(1 + \frac{\bar{h}_n p_n}{N_0}\big), \tag{14}$$

where B is the bandwidth, N_0 is the background noise, p_n is the transmission power, and $\bar{h}_n$ is the average channel gain of the UE n during the training time of FEDL. Denote the fraction of communication time allocated to UE n by τ_n, and the data size (in nats) of w_n and $\nabla F_n(w_n)$ by s_n. Because the dimension of vectors w_n and $\nabla F_n(w_n)$ is fixed, we assume that their sizes are constant throughout the FEDL learning. Then the transmission rate of each UE n is

$$r_n = s_n/\tau_n, \tag{15}$$

which is shown to be the most energy-efficient transmission policy [32]. Thus, to transmit s_n within a time duration τ_n, the UE n's energy consumption is

$$E_{n,co} = \tau_n\, p_n(s_n/\tau_n), \tag{16}$$

where the power function is

$$p_n(s_n/\tau_n) := \frac{N_0}{\bar{h}_n}\Big(e^{\frac{s_n/\tau_n}{B}} - 1\Big) \tag{17}$$

according to (14) and (15). We denote the vector of τ_n by $\tau \in \mathbb{R}^n$.

Define the total energy consumption of all UEs for each global round by E_g, which is expressed as follows:

$$E_g := \sum\nolimits_{n=1}^{N} E_{n,co} + K_l\, E_{n,cp}.$$

B. Problem Formulation

We consider an optimization problem, abusing the same name FEDL, as follows

$$\begin{aligned}
&\underset{f,\tau,\theta,\eta,T_{co},T_{cp}}{\text{minimize}} \; K_g\big(E_g + \kappa\, T_g\big)\\
&\text{subject to } \sum\nolimits_{n=1}^{N} \tau_n \le T_{co}, && (18)\\
&\qquad \max_n \frac{c_n D_n}{f_n} = T_{cp}, && (19)\\
&\qquad f_n^{min} \le f_n \le f_n^{max}, \quad \forall n \in \mathcal{N}, && (20)\\
&\qquad p_n^{min} \le p_n(s_n/\tau_n) \le p_n^{max}, \quad \forall n \in \mathcal{N}, && (21)\\
&\qquad 0 \le \theta \le 1. && (22)
\end{aligned}$$

Minimize both UEs' energy consumption and the FL time are conflicting. For example, the UEs can save the energy by setting the lowest frequency level all the time, but this will certainly increase the training time. Therefore, to strike the balance between energy cost and training time, the weight κ (Joules/second), used in the objective as an amount of additional energy cost that FEDL is willing to bear for one unit of training time to be reduced, captures the Pareto-optimal tradeoff between the UEs' energy cost and the FL time. For example, when most of the UEs are plugged in, then UE energy is not the main concern, thus κ can be large. According to optimization theory, $1/\kappa$ also plays the role of a Lagrange multiplier for a "hard constraint" on UE energy [29].

While constraint (18) captures the time-sharing uplink transmission of UEs, constraint (19) defines that the computing time in one local round is determined by the "bottleneck" UE (e.g., with large data size and low CPU frequency). The feasible regions of CPU-frequency and transmit power of UEs are imposed by constraints (20) and (21), respectively. We note that (20) and (21) also capture the heterogeneity of UEs with different types of CPU and transmit chipsets. The last constraint restricts the feasible range of the local accuracy.

C. Solutions to FEDL

We see that FEDL is non-convex due to the constraint (19) and several products of two functions in the objective function. However, in this section we will characterize FEDL's solution by decomposing it into multiple simpler sub-problems.

We consider the first case when θ and η are fixed, then FEDL can be decomposed into two sub-problems as follows:

$$\begin{aligned}
&\textsf{SUB1}\colon \underset{f,T_{cp}}{\text{minimize}} \sum\nolimits_{n=1}^{N} E_{n,cp} + \kappa T_{cp}\\
&\qquad \text{subject to } \frac{c_n D_n}{f_n} \le T_{cp}, \quad \forall n \in \mathcal{N},\\
&\qquad\qquad f_n^{min} \le f_n \le f_n^{max}, \quad \forall n \in \mathcal{N}. \quad (23)
\end{aligned}$$

$$\begin{aligned}
&\textsf{SUB2}\colon \min_{\tau,T_{co}} \sum\nolimits_{n=1}^{N} E_{n,co} + \kappa T_{co}\\
&\qquad \text{s.t. } \sum\nolimits_{n=1}^{N} \tau_n \le T_{co}, && (24)\\
&\qquad\quad p_n^{min} \le p_n(s_n/\tau_n) \le p_n^{max}, \quad \forall n. && (25)
\end{aligned}$$

While SUB1 is a CPU-cycle control problem for the computation time and energy minimization, SUB2 can be considered as an uplink power control to determine the UEs' fraction of time sharing to minimize the UEs energy and communication time. We note that the constraint (19) of FEDL is replaced by an equivalent one (23) in SUB1. We can consider T_{cp} and T_{co} as virtual deadlines for UEs to perform their computation and communication updates, respectively. It can be observed that both SUB1 and SUB2 are convex problems.

1) SUB1 Solution: We first propose Algorithm 2 in order to categorize UEs into one of three groups: $\mathcal{N}_1$ is a group of "bottleneck" UEs that always run its maximum frequency; $\mathcal{N}_2$ is the group of "strong" UEs which can finish their tasks before the computational virtual deadline even with the minimum frequency; and $\mathcal{N}_3$ is the group of UEs having the optimal frequency inside the interior of their feasible sets.

Algorithm 2 Finding $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$ in Lemma 2

Sort UEs such that $\frac{c_1 D_1}{f_1^{min}} \le \frac{c_2 D_2}{f_2^{min}} \ldots \le \frac{c_N D_N}{f_N^{min}}$
Input: $\mathcal{N}_1 = \emptyset$, $\mathcal{N}_2 = \emptyset$, $\mathcal{N}_3 = \mathcal{N}$, $T_{\mathcal{N}_3}$ in (28)
for $i = 1$ to N **do**
  if $\max_{n\in\mathcal{N}} \frac{c_n D_n}{f_n^{max}} \ge T_{\mathcal{N}_3} > 0$ and $\mathcal{N}_1 == \emptyset$ **then**
    $\mathcal{N}_1 = \mathcal{N}_1 \cup \big\{m : \frac{c_m D_m}{f_m^{max}} = \max_{n\in\mathcal{N}} \frac{c_n D_n}{f_n^{max}}\big\}$
    $\mathcal{N}_3 = \mathcal{N}_3 \setminus \mathcal{N}_1$ and update $T_{\mathcal{N}_3}$ in (28)
  if $\frac{c_i D_i}{f_i^{min}} \le T_{\mathcal{N}_3}$ **then**
    $\mathcal{N}_2 = \mathcal{N}_2 \cup \{i\}$
    $\mathcal{N}_3 = \mathcal{N}_3 \setminus \{i\}$ and update $T_{\mathcal{N}_3}$ in (28)

Lemma 2: The optimal solution to SUB1 is as follows

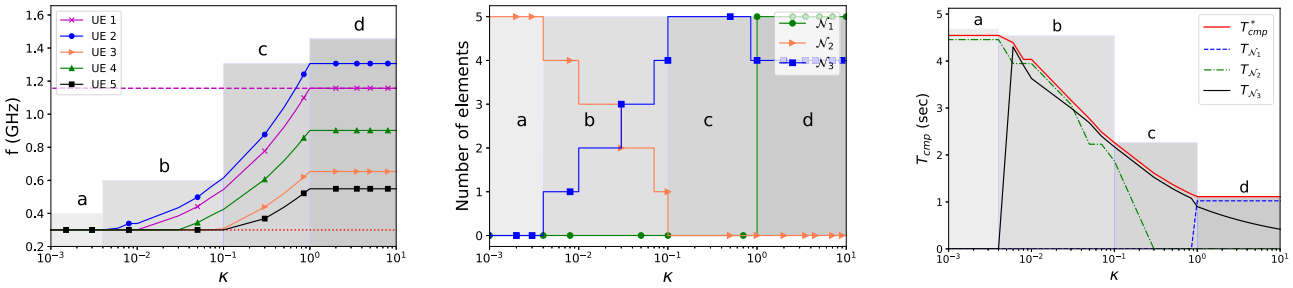
$$f_n^* = \begin{cases} f_n^{max}, & \forall n \in \mathcal{N}_1,\\ f_n^{min}, & \forall n \in \mathcal{N}_2,\\ \dfrac{c_n D_n}{T_{cp}^*}, & \forall n \in \mathcal{N}_3, \end{cases} \quad (26)$$

$$T_{cp}^* = \max\big\{T_{\mathcal{N}_1}, T_{\mathcal{N}_2}, T_{\mathcal{N}_3}\big\}, \quad (27)$$

where $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3 \subseteq \mathcal{N}$ are three subsets of UEs produced by Algorithm 2 and

$$\begin{aligned}
T_{\mathcal{N}_1} &= \max_{n\in\mathcal{N}} \frac{c_n D_n}{f_n^{max}},\\
T_{\mathcal{N}_2} &= \max_{n\in\mathcal{N}_2} \frac{c_n D_n}{f_n^{min}},\\
T_{\mathcal{N}_3} &= \left(\frac{\sum_{n\in\mathcal{N}_3} \alpha_n (c_n D_n)^3}{\kappa}\right)^{1/3}. \quad (28)
\end{aligned}$$

From Lemma 2, first, we see that the optimal solution depends not only on the existence of these subsets, but also on their virtual deadlines $T_{\mathcal{N}_1}$, $T_{\mathcal{N}_2}$, and $T_{\mathcal{N}_3}$, in which the longest of them will determine the optimal virtual deadline T_{cp}^*. Second, from (26), the optimal frequency of each UE will



(a) Optimal CPU frequency of each UE. (b) Three subsets outputted by Alg. 2. (c) Optimal computation time.

Fig. 1. Solution to SUB1 with five UEs. For wireless communication model, the UE channel gains follow the exponential distribution with the mean $g_0(d_0/d)^4$ where $g_0 = -40$ dB and the reference distance $d_0 = 1$ m. The distance between these devices and the wireless access point is uniformly distributed between 2 and 50 m. In addition, $B = 1$ MHz, $\sigma = 10^{-10}$ W, the transmission power of devices are limited from 0.2 to 1 W. For UE computation model, we set the training size D_n of each UE as uniform distribution in $5-10$ MB, c_n is uniformly distributed in $10-30$ cycles/bit, f_n^{max} is uniformly distributed in $1.0-2.0$ GHz, $f_n^{min} = 0.3$ GHz. Furthermore, $\alpha = 2\times10^{-28}$ and the UE update size $s_n = 25,000$ nats (≈4.5 KB).

depend on both T^*_{cp} and the subset it belongs to. We note that depending on κ, some of the three sets (not all) are possibly empty sets, and by default $T_{\mathcal{N}_i} = 0$ if $\mathcal{N}_i$ is an empty set, $i = 1, 2, 3$. Next, by varying κ, we observe the following special cases.

Corollary 2: The optimal solution to SUB1 can be divided into four regions as follows.

a) $\kappa \le \min_{n\in\mathcal{N}} \alpha_n (f_n^{min})^3$:
$\mathcal{N}_1$ and $\mathcal{N}_3$ are empty sets. Thus, $\mathcal{N}_2 = \mathcal{N}$, $T^*_{co} = T_{\mathcal{N}_2} = \max_{n\in\mathcal{N}} \frac{c_n D_n}{f_n^{min}}$, and $f_n^* = f_n^{min}, \forall n \in \mathcal{N}$.

b) $\min_{n\in\mathcal{N}} \alpha_n (f_n^{min})^3 < \kappa \le \left(\max_{n\in\mathcal{N}_2} \frac{c_n D_n}{f_n^{min}}\right)^3$:
$\mathcal{N}_2$ and $\mathcal{N}_3$ are non-empty sets, whereas $\mathcal{N}_1$ is empty. Thus, $T^*_{cp} = \max\{T_{\mathcal{N}_2}, T_{\mathcal{N}_3}\}$, and $f_n^* = \max\{\frac{c_n D_n}{T^*_{cp}}, f_n^{min}\}, \forall n \in \mathcal{N}$.

c) $\left(\max_{n\in\mathcal{N}_2} \frac{c_n D_n}{f_n^{min}}\right)^3 < \kappa \le \frac{\sum_{n\in\mathcal{N}_3} \alpha_n \left(c_n D_n\right)^3}{\left(\max_{n\in\mathcal{N}} \frac{c_n D_n}{f_n^{max}}\right)^3}$:
$\mathcal{N}_1$ and $\mathcal{N}_2$ are empty sets. Thus $\mathcal{N}_3 = \mathcal{N}$, $T^*_{cp} = T_{\mathcal{N}_3}$, and $f_n^* = \frac{c_n D_n}{T_{\mathcal{N}_3}}, \forall n \in \mathcal{N}$.

d) $\kappa > \frac{\sum_{n\in\mathcal{N}_3} \alpha_n \left(c_n D_n\right)^3}{\left(\max_{n\in\mathcal{N}} \frac{c_n D_n}{f_n^{max}}\right)^3}$:
$\mathcal{N}_1$ is non-empty. Thus $T^*_{cp} = T_{\mathcal{N}_1}$, and

$$f_n^* = \begin{cases} f_n^{max}, & \forall n \in \mathcal{N}_1, \\ \max\{\dfrac{c_n D_n}{T_{\mathcal{N}_1}}, f_n^{min}\}, & \forall n \in \mathcal{N} \setminus \mathcal{N}_1. \end{cases} \tag{29}$$

We illustrate Corollary 2 in Fig. 1 with four regions[1] as follows.

a) Very low κ (i.e., $\kappa \le 0.004$): Designed for solely energy minimization. In this region, all UE runs their CPU at the lowest cycle frequency f_n^{min}, thus T^*_{cp} is determined by the last UEs that finish their computation with their minimum frequency.

b) Low κ (i.e., $0.004 \le \kappa \le 0.1$): Designed for prioritized energy minimization. This region contains UEs of both $\mathcal{N}_2$ and $\mathcal{N}_3$. T^*_{cp} is governed by which subset has higher virtual computation deadline, which also determines the optimal CPU-cycle frequency of $\mathcal{N}_3$. Other UEs with light-loaded data, if exist, can run at the most energy-saving mode f_n^{min} yet still finish their task before T^*_{cp} (i.e., $\mathcal{N}_2$).

c) Medium κ (i.e., $0.1 \le \kappa \le 1$): Designed for balancing computation time and energy minimization. All UEs belong to $\mathcal{N}_3$ with their optimal CPU-cycle frequency strictly inside the feasible set.

d) High κ (i.e., $\kappa \ge 1$): Designed for prioritized computation time minimization. High value κ can ensure the existence of $\mathcal{N}_1$, consisting the most "bottleneck" UEs (i.e., heavy-loaded data and/or low f_n^{max}) that runs their maximum CPU-cycle in (29) (top) and thus determines the optimal computation time T^*_{cp}. The other "non-bottleneck" UEs either (i) adjust a "right" CPU-cycle to save the energy yet still maintain their computing time the same as T^*_{cp} (i.e., $\mathcal{N}_3$), or (ii) can finish the computation with minimum frequency before the "bottleneck" UEs (i.e., $\mathcal{N}_2$) as in (29) (bottom).

2) SUB2 Solution: Before characterizing the solution to SUB2, from (17) and (25), we first define two bounded values for τ_n as follows

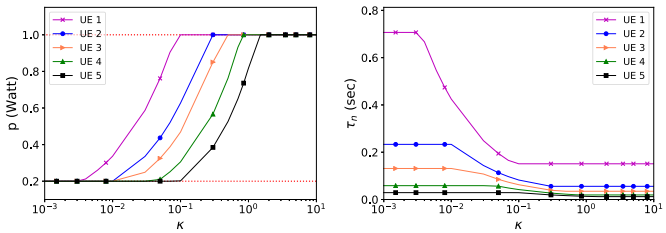
$$\tau_n^{max} = \frac{s_n}{B\ln(\bar{h}_n N_0^{-1} p_n^{min} + 1)},$$
$$\tau_n^{min} = \frac{s_n}{B\ln(\bar{h}_n N_0^{-1} p_n^{max} + 1)},$$

which are the maximum and minimum possible fractions of T_{co} that UE n can achieve by transmitting with its minimum and maximum power, respectively. We also define a new function $g_n : \mathbb{R} \to \mathbb{R}$ as

$$g_n(\kappa) = \frac{s_n/B}{1 + W\left(\frac{\kappa N_0^{-1}\bar{h}_n - 1}{e}\right)},$$

where $W(\cdot)$ is the Lambert W-function. We can consider $g_n(\cdot)$ as an indirect "power control" function that helps UE n control the amount of time it should transmit an amount of data s_n by adjusting the power based on the weight κ. This function is strictly decreasing (thus its inverse function $g_n^{-1}(\cdot)$ exists) reflecting that when we put more priotity on minimizing the communication time (i.e., high κ), UE n should raise the power to finish its transmission with less time (i.e., low τ_n).

[1]All closed-form solutions are also verified by the solver IPOPT [33].



(a) UEs' optimal transmission power. (b) UEs' optimal transmission time .

Fig. 2. The solution to SUB2 with five UEs. The numerical setting is the same as that of Fig. 1.

Lemma 3: The solution to SUB2 is as follows

a) If $\kappa \leq g_n^{-1}(\tau_n^{max})$, *then*

$$\tau_n^* = \tau_n^{max}$$

b) If $g_n^{-1}(\tau_n^{max}) < \kappa < g_n^{-1}(\tau_n^{min})$, *then*

$$\tau_n^{min} < \tau_n^* = g_n(\kappa) < \tau_n^{max}$$

c) If $\kappa \geq g_n^{-1}(\tau_n^{min})$, *then*

$$\tau_n^* = \tau_n^{min},$$

and $T_{co}^* = \sum_{n=1}^N \tau_n^*$.

This lemma can be explained through the lens of network economics. If we interpret the FEDL system as the buyer and UEs as sellers with the UE powers as commodities, then the inverse function $g_n^{-1}(\cdot)$ is interpreted as the price of energy that UE n is willing to accept to provide power service for FEDL to reduce the training time. There are two properties of this function: (i) the price increases with respect to UE power, and (ii) the price sensitivity depends on UEs characteristics, e.g., UEs with better channel quality can have lower price, whereas UEs with larger data size s_n will have higher price. Thus, each UE n will compare its energy price $g_n^{-1}(\cdot)$ with the "offer" price κ by the system to decide how much power it is willing to "sell". Then, there are three cases corresponding to the solutions to SUB2.

a) Low offer: If the offer price κ is lower than the minimum price request $g_n^{-1}(\tau_n^{max})$, UE n will sell its lowest service by transmitting with the minimum power p_n^{min}.
b) Medium offer: If the offer price κ is within the range of an acceptable price range, UE n will find a power level such that the corresponding energy price will match the offer price.
c) High offer: If the offer price κ is higher than the maximum price request $g_n^{-1}(\tau_n^{min})$, UE n will sell its highest service by transmitting with the maximum power p_n^{max}.

Lemma 3 is further illustrated in Fig. 2, showing how the solution to SUB2 varies with respect to κ. It is observed from this figure that due to the UE heterogeneity of channel gain, $\kappa = 0.1$ is a medium offer to UEs 2, 3, and 4, but a high offer to UE 1, and low offer to UE 5.

While SUB1 and SUB2 solutions share the same threshold-based dependence, we observe their differences as follows. In SUB1 solution, the optimal CPU-cycle frequency of UE n depends on the optimal T_{cp}^*, which in turn depends on the loads (i.e., $\frac{c_n D_n}{f_n}, \forall n$) of all UEs. Thus all UE load information is required for the computation phase. On the other hand, in SUB2 solution, each UE n can independently choose its optimal power by comparing its price function $g_n^{-1}(\cdot)$ with κ so that collecting UE information is not needed. The reason is that the synchronization of computation time in constraint (23) of SUB1 requires all UE loads, whereas the UEs' time-sharing constraint (24) of SUB2 can be decoupled by comparing with the fixed "offer" price κ.

TABLE I

THE SOLUTION TO SUB3 WITH FIVE UES. THE NUMERICAL SETTING IS THE SAME AS THAT OF FIG. 1

ρ	κ	θ^*	Θ	η^*
1.4/2/5	0.1	.033/.015/.002	.094/.042/.003	.253/.177/.036
	1	.035/.016/.002	.092/.041/.003	.253/.177/.036
	10	.035/.016/.002	.092/.041/.003	.253/.177/.036

3) SUB3 Solution: We observe that the solutions to SUB1 and SUB2 have no dependence on θ so that the optimal T_{co}^*, T_{cp}^*, f^*, τ^*, and thus the corresponding optimal energy values, denoted by $E_{n,cp}^*$ and $E_{n,cp}^*$, can be determined based on κ according to Lemmas 2 and 3. However, these solutions will affect to the third sub-problem of FEDL, as will be shown in what follows.

$$\text{SUB3}: \quad \underset{\theta,\eta>0}{\text{minimize}} \ \frac{1}{\Theta}\Big(\sum\nolimits_{n=1}^N E_{n,co}^* + K_l\, E_{n,cp}^* + \kappa\big(T_{co}^* + K_l\, T_{cp}^*\big)\Big)$$
$$\text{subject to } 0 < \theta < 1, 0 < \Theta < 1.$$

SUB3 is unfortunately non-convex. However, since there are only two variables to optimize, we can employ numerical methods to find the optimal solution. The numerical results in Table I show that the solution θ^* and η^* to SUB3 decreases when ρ increases, which makes Θ decreases, as explained by the results of Theorem 1. Also we observe that κ as more effect to the solution to SUB3 when ρ is small.

4) FEDL Solution: Since we can obtain the stationary points of SUB3 using Successive Convex Approximation techniques such as NOVA [34], then we have:

Theorem 2: The combined solutions to three sub-problems SUB1, SUB2, and SUB3 are stationary points of FEDL.

The proof of this theorem is straightforward. The idea is to use the KKT condition to find the stationary points of FEDL. Then we can decompose the KKT condition into three independent groups of equations (i.e., no coupling variables between them), in which the first two groups matches exactly to the KKT conditions of SUB1 and SUB2 that can be solved by closed-form solutions as in Lemmas 2, 3, and the last group for SUB3 is solved by numerical methods.

We then have some discussions on the combined solution to FEDL. First, we see that SUB1 and SUB2 solutions can be characterized independently, which can be explained that each UE often has two separate processors: one CPU for mobile applications and another baseband processor for radio control function. Second, neither SUB1 nor SUB2 depends on θ because the communication phase in SUB2 is clearly not affected by the local accuracy, whereas SUB2 considers the computation cost in one local round. However, the solutions to

TABLE II
SUMMARY OF THE SUB-PROBLEM COMPLEXITY

SUB1	SUB2	SUB3
$O(N^2)$	$O(1)$	$O(N)$

SUB1 and SUB2, which can reveal how much communication cost is more expensive than computation cost, are decisive factors to determine the optimal level of local accuracy. Therefore, we can sequentially solve SUB1 and SUB2 first, then SUB3 to achieve the solutions to FEDL. We also summarize the complexities in the following table:

VI. EXPERIMENTS

This section will validate the FEDL's learning performance in a heterogeneous network. The experimental results show that FEDL gains performance improvement from the vanilla FedAvg [4] in terms of training loss convergence rate and test accuracy in various settings. All codes and data are published on GitHub [35].

Experimental settings: In our setting, the performance of FEDL is examined by both classification and regression tasks. The regression task uses linear regression model with mean square error loss function on a synthetic dataset while the classification task uses multinomial logistic regression model with cross-entropy error loss function on real federated datasets (MNIST [36], FEMNIST [37]). The loss function for each UE is given below:

1) Mean square error loss for linear regression (synthetic dataset):

$$F_n(w) = \frac{1}{D_n} \sum_{(x_i,y_i)\in\mathcal{D}_n} (\langle x_i, w\rangle - y_i)^2.$$

2) Cross-entropy loss for multinomial logistic regression with C class labels (MNIST and FEMNIST datasets):

$$F_n(w) = \frac{-1}{D_n}\left[\sum_{i=1}^{D_n}\sum_{c=1}^{C} 1_{\{y_i=c\}} \log \frac{\exp(\langle x_i, w_c\rangle)}{\sum_{j=1}^{C}\exp(\langle x_i, w_j\rangle)}\right] + \frac{\beta}{2}\sum_{c=1}^{C}\|w_c\|^2.$$

We consider $N = 100$ UEs. To verify that FEDL also works with UE subset sampling, we allow FEDL to randomly sample a number of subset of UEs, denoted by S, following a uniform distribution as in FedAvg in each global iteration. In order to generate datasets capturing the heterogeneous nature of FL, all datasets have different sample sizes based on the power law in [9]. In MNIST, each user contains three of the total of ten labels. FEMNIST is built similar to [9] by partitioning the data in Extended MNIST [38]. For synthetic data, to allow for the non-iid setting, each user's data has the dimension $d = 40$. We control the value of ρ by using the data generation method similar to that in [39], in which user i has data drawn from $\mathcal{N}(0, \sigma_i\Sigma)$ where $\sigma_i \sim \mathcal{U}(1, 10)$. Here Σ is a diagonal covariance matrix with $\Sigma_{ii} = i^{-p}, i \in [1, d]$ and $p = \frac{\log(\rho)}{\log(d)}$, where ρ is considered as multiplicative inverse for the minimum covariance value of Σ. The number of data samples of each UE is in the ranges $[55, 3012]$, $[504, 1056]$, and $[500, 5326]$ for MNIST, FEMNIST, and Synthetic, respectively. All datasets are split randomly with 75% for training and 25% for testing. Each experiment is run at least 10 times and the average results are reported. We summarized all parameters used for the experiments in Table. III.

TABLE III
EXPERIMENT PARAMETERS

Parameters	Description
ρ	Condition number of $F_n(\cdot)'s$ Hessian Matrix, $\rho = \frac{L}{\beta}$
K_g	Global rounds for global model update
K_l	Local rounds for local model update
η	Hyper-learning rate
h_k	Local learning rate
B	Batch Size ($B = \infty$ means full batch size for GD)

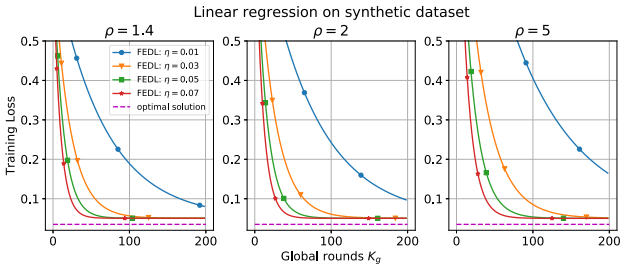


Fig. 3. Effect of η on the convergence of FEDL. Training processes use full-batch gradient descent, full devices participation ($N = S = 100$ UEs), $K_g = 200$, and $K_l = 20$.

Effect of the hyper-learning rate on FEDL's convergence: We first verify the theoretical finding by predetermining the value of ρ and observing the impact of changing η on the convergence of FEDL using a synthetic dataset. In Fig. 3, we examine four different values of ρ. As can be seen in the figure, with all value of ρ, there were exist small enough values of η that allow FEDL to converge. We also observe that using the larger value of η makes FEDL converge faster. In addition, even if FEDL allows UEs to solve their local problems approximately, the experiment shows that the gap between the optimal solution and our approach in Fig. 3 is negligible. It is noted that the optimal solution is obtained by solving directly the global loss function (1) as we consider the local loss function at each UE is mean square error loss.

Effect of different gradient descent algorithms on FEDL's performance: As UEs are allowed to use different gradient descent methods to minimize the local problem (2), the convergence of FEDL can be evaluated on different optimization algorithms: GD and mini-batch SGD by changing the configuration of the batch size during the local training process. Although our analysis results are based on GD, we also monitor the behavior of FEDL using SGD in the experiments for the comparison. While a full batch size is applied for GD, mini-batch SGD is trained with a batch size of 20 and 40 samples. We conducted a grid search on h_k to find the value allowing FEDL and FedAvg to obtain the best performance in terms of accuracy and stability. Fig. 4 demonstrates that FEDL outperforms FedAvg on all batch size settings (the improvement in terms of testing accuracy and training loss are approximately 1.3% and 9.1% respectively for the batch size 20, 0.7% and -0.2% for the batch size 40,

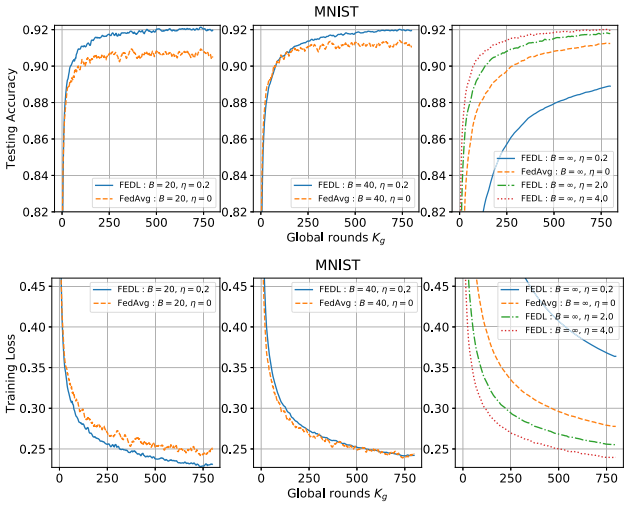


Fig. 4. Effect of different batch-size with fixed value of $K_l = 20$, $K_g = 800$, $N = 100$, $S = 10$, ($B = \infty$ means full batch size) on FEDL's performance.

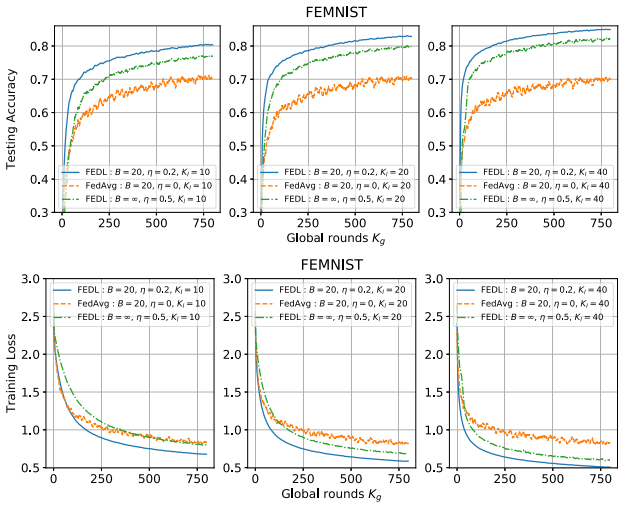


Fig. 5. Effect of increasing local computation time on the convergence of FEDL ($N = 100$, $S = 10$, $K_g = 800$).

and 0.8% and 14% for the full batch size). Besides, FEDL is more stable than FedAvg when the small number of devices is sub-sampling randomly during the training process. Even though using larger batch size benefits the stability of both FedAvg and FEDL, very large batch size can make the convergence of FEDL slow. However, increasing the value of η allows speeding up the convergence of FEDL in case of GD.

Effect of increasing local computation on convergence time: In order to validate the performance of FEDL on a different value of local updates K_l, in the Fig. 5, we use both mini-batch SGD algorithm with the fixed batch size of 20 and GD for the local update and increase K_l from 10 to 40. For all K_l, even when the learning rate of FedAvg is tuned carefully, FEDL in all batch size settings achieve a significant performance gap over FedAvg in terms of training loss and testing accuracy. Also in this experiment, FEDL using minibatch outperforms FEDL using GD for FEMNIST dataset. While larger K_l does not show the improvement on the convergence of FedAvg, the rise of K_l has an appreciably positive impact on the convergence time FEDL.

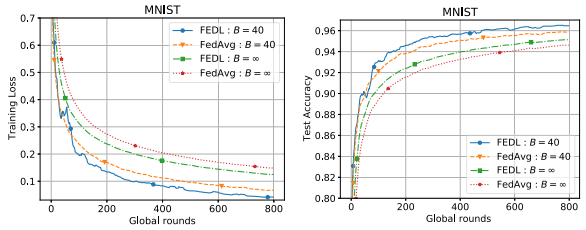
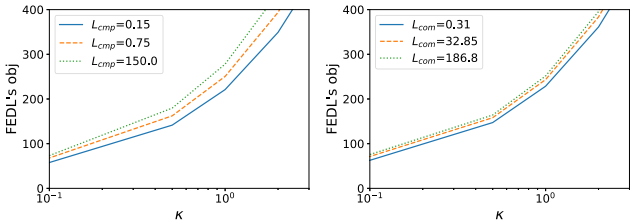


Fig. 6. FEDL's performance on non-convex (MNIST dataset).



(a) Impact of L_{cp} (b) Impact of L_{co}

Fig. 7. Impact of UE heterogeneity on FEDL.

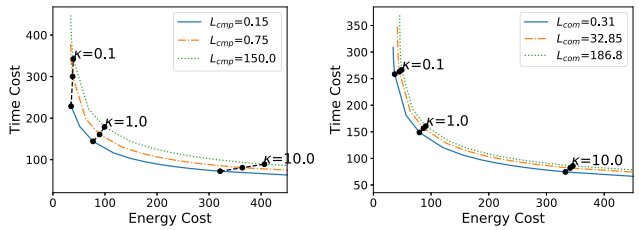
However, the larger K_l requires higher local computation at UEs, which costs the EU's energy consumption.

FEDL's performance on non-convex problem: Even though the convergence of FEDL is only applicable to strongly convex and smooth loss functions in theory, we will show that FEDL also empirically works well in non-convex case. We consider a simple non-convex model that has a two-layer deep neural network (DNN) with one hidden layer of size 100 using ReLU activation, an output layer using a softmax function, and the negative log-likelihood loss function. In Fig. 6, we see that by using a more complexed model, both FEDL and FedAvg achieve higher accuracy than the strongly convex case. Although FEDL still outperforms FedAvg in case of DNN, the performance improvement is negligible compared to the strongly convex case, and FEDL become less stable when using small mini-batch size.

VII. Numerical Results

In this section, both the communication and computation models follow the same setting as in Fig. 1, except the number of UEs is increased to 50, and all UEs have the same $f_n^{max} = 2.0$ GHz and $c_n = 20$ cycles/bit. Furthermore, we define two new parameters, addressing the UE heterogeneity regarding to computation and communication phases in FEDL, respectively, as $L_{cp} = \frac{\max_{n\in\mathcal{N}} \frac{c_n D_n}{f_n^{max}}}{\min_{n\in\mathcal{N}} \frac{c_n D_n}{f_n^{min}}}$ and $L_{co} = \frac{\max_{n\in\mathcal{N}} \tau_n^{min}}{\min_{n\in\mathcal{N}} \tau_n^{max}}$. We see that higher values of L_{cp} and L_{co} indicate higher levels of UE heterogeneity. The level of heterogeneity is controlled by two different settings. To vary L_{cp}, the training size D_n is generated with the fraction $\frac{D^{min}}{D^{max}} \in \{1, 0.2, 0.001\}$ but the average UE data size is kept at the same value 7.5 MB for varying values of L_{cp}. On the other hand, to vary L_{co}, the distance between these devices and the edge server is generated such that $\frac{d^{min}}{d^{max}} \in \{1., 0.2, 0.001\}$ but the average distance of all UEs is maintained at 26 m.

We first examine the impact of UE heterogeneity. We observe that the high level of UE heterogeneity has negative impact on the FEDL system, as illustrated in Figs. 7a and 7b, such that the total cost is increased with higher value



(a) Impact of L_{cp} and κ. (b) Impact of L_{co} and κ.

Fig. 8. Pareto-optimal points of FEDL.

of L_{cp} and L_{co} respectively. We next illustrate the Pareto curve in Fig. 8. This curve shows the trade-off between the conflicting goals of minimizing the time cost $K(\theta)T_g$ and energy cost $K(\theta)E_g$, in which we can decrease one type of cost yet with the expense of increasing the other one. This figure also shows that the Pareto curve of FEDL is more efficient when the system has low level of UE heterogeneity (i.e., small L_{cp} and/or L_{co}).

VIII. Conclusions

In this paper, we studied FL, a learning scheme in which the training model is distributed to participating UEs performing training over their local data. Although FL shows vital advantages in data privacy, the heterogeneity across users' data and UEs' characteristics are still challenging problems. We proposed an effective algorithm without the i.i.d. UEs' data assumption for strongly convex and smooth FL's problems and then characterize the algorithm's convergence. For the wireless resource allocation problem, we embedded the proposed FL algorithm in wireless networks which considers the trace-offs not only between computation and communication latencies but also the FL time and UE energy consumption. Despite the non-convex nature of this problem, we decomposed it into three sub-problems with convex structure before analyzing their closed-form solutions and quantitative insights into problem design. We then verified the theoretical findings of the new algorithm by experiments on Tensoflow with several datasets, and the wireless resource allocation sub-problems by extensive numerical results. In addition to validating the theoretical convergence, our experiments also showed that the proposed algorithm can boost the convergence speed compared to an existing baseline approach.

References

[1] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, Paris, France, Apr. 2019, pp. 1387–1395.

[2] D. Reinsel, J. Gantz, and J. Rydning, "The digitization of the world from edge to core," IDC, Framingham, MA, USA, White Paper Doc# US44413318, 2018, p. 28. [Online]. Available: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf

[3] *Free Community-based GPS, Maps & Traffic Navigation App | Waze*. Accessed: Sep. 1, 2019. [Online]. Available: https://www.waze.com/

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (PMLR)*, vol. 54, 2017, pp. 1273–1282.

[5] A. Williams, "What does AI in a phone really mean?," Oct. 22, 2018. Accessed: Sep. 1, 2019. [Online]. Available: https://www.techradar.com/news/what-does-ai-in-a-phone-really-mean

[6] J. Konečný, H. Brendan McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*. [Online]. Available: http://arxiv.org/abs/1610.02527

[7] 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.

[8] V. Smith, S. Forte, C. Ma, M. Takáč, M. I. Jordan, and M. Jaggi, "CoCoA: A general framework for communication-efficient distributed optimization," *J. Mach. Learn. Res.*, vol. 18, no. 230, pp. 1–49, 2018.

[9] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. 1st Adapt.Multitask Learn., ICML Workshop*, Long Beach, CA, USA, 2019, p. 16.

[10] C. Ma *et al.*, "Distributed optimization with arbitrary local solvers," *Optim. Methods Softw.*, vol. 32, no. 4, pp. 813–848, Jul. 2017.

[11] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *Proc. ICML*, Beijing, China, vol. 2014, pp. II–1000–II–1008.

[12] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," 2018, *arXiv:1808.07576*. [Online]. Available: http://arxiv.org/abs/1808.07576

[13] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019. [Online]. Available: https://iclr.cc/Conferences/2019/Schedule?showEvent=979

[14] F. Zhou and G. Cong, "On the convergence properties of a K-step averaging stochastic gradient descent algorithm for nonconvex optimization," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Stockholm, Sweden, Jul. 2018, pp. 3219–3227.

[15] J. Konečný, H. Brendan McMahan, F. X. Yu, P. Richtárik, A. Theertha Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*. [Online]. Available: http://arxiv.org/abs/1610.05492

[16] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proc. NeurIPS*, Long Beach, CA, USA, 2017, pp. 4427–4437.

[17] C. T. Dinh, N. H. Tran, T. D. Nguyen, W. Bao, A. Y. Zomaya, and B. B. Zhou, "Federated learning with proximal stochastic variance reduced gradient algorithms," in *Proc. 49th Int. Conf. Parallel Process. (ICPP)*, Edmonton, AB, Canada, Aug. 2020, pp. 1–11.

[18] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized federated learning with Moreau envelopes," in *34th Conf. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, BC, Canada, 2020.

[19] M. M. Amiri and D. Gunduz, "Over-the-air machine learning at the wireless edge," in *Proc. IEEE 20th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jul. 2019, pp. 1–5.

[20] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Proc. NeurIPS*, 2018, pp. 7663–7673.

[21] T. Thanh Vu, D. Trong Ngo, N. H. Tran, H. Q. Ngo, M. N. Dao, and R. H. Middleton, "Cell-free massive MIMO for wireless federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6377–6392, Oct. 2020.

[22] L. U. Khan *et al.*, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Commun. Mag.*, to be published. [Online]. Available: https://arxiv.org/abs/1911.05642

[23] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, A. Manzoor, an C. S. Hong, "A crowdsourcing framework for on-device federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3241–3256, May 2020.

[24] M. Chen, Z. Yang, W. Saad, C. Yin, H. Vincent Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," 2019, *arXiv:1909.07972*. [Online]. Available: http://arxiv.org/abs/1909.07972

[25] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "ATOMO: Communication-efficient learning via atomic sparsification," in *Proc. NeurIPS*, 2018, pp. 9850–9861.

[26] H. Zhang, J. Li, K. Kara, D. Alistarh, J. Liu, and C. Zhang, "ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2017, pp. 4035–4043.

[27] S. J. Reddi, J. Konečný, P. Richtárik, B. Póczós, and A. Smola, "AIDE: Fast and communication efficient distributed optimization," 2016, *arXiv:1608.06879*. [Online]. Available: http://arxiv.org/abs/1608.06879

[28] Y. Nesterov, *Lectures Convex Optimization*, vol. 137. Springer, 2018,

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, Mar. 2004.
[30] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX HotCloud*, Berkeley, CA, USA, 2010, p. 4.
[31] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *J. VLSI Signal Process. Syst.*, vol. 13, nos. 2–3, pp. 203–221, Aug. 1996.
[32] B. Prabhakar, E. U. Biyikoglu, and A. El Gamal, "Energy-efficient transmission over a wireless link via lazy packet scheduling," in *Proc. IEEE Conf. Comput. Commun. 20th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, vol. 1, Apr. 2001, pp. 386–394.
[33] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.
[34] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and distributed methods for constrained nonconvex optimization—Part I: theory," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 1929–1944, Apr. 2017.
[35] C. T. Dinh. Accessed: Sep. 2020. [Online]. Available: https://github.com/CharlieDinh/FEDL_Pytorch
[36] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[37] S. Caldas *et al.*, "LEAF: A benchmark for federated settings," 2018, *arXiv:1812.01097*. [Online]. Available: http://arxiv.org/abs/1812.01097
[38] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 2921–2926.
[39] X. Li, W. Yang, S. Wang, and Z. Zhang, "Communication-efficient local decentralized SGD methods," 2019, *arXiv:1910.09126*. [Online]. Available: http://arxiv.org/abs/1910.09126

Canh T. Dinh received the B.E. degree in electronics and telecommunication from the Ha Noi University of Science and Technology, Ha Noi City, Vietnam, in 2015, and the master's degree in data science from the Université Grenoble Alpes, Grenoble, France, in 2019. He is currently pursuing the Ph.D. degree in computer science with The University of Sydney, Sydney, Australia. His supervisor is Dr. Nguyen H. Tran. His research interests include distributed computing and federated learning.

Nguyen H. Tran (Senior Member, IEEE) received the B.S. and Ph.D. degrees in electrical and computer engineering from the HCMC University of Technology and Kyung Hee University, in 2005 and 2011, respectively. He was an Assistant Professor with the Department of Computer Science and Engineering, Kyung Hee University, from 2012 to 2017. Since 2018, he has been with the School of Computer Science, The University of Sydney, where he is currently a Senior Lecturer. His research interests include distributed computing, machine learning, and networking. He received the Best KHU Thesis Award in engineering in 2011 and several best paper awards, including the IEEE ICC 2016 and ACM MSWiM 2019. He receives the Korea NRF Funding for Basic Science and Research 2016–2023 and the ARC Discovery Project 2020–2023. He was the Editor of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING from 2016 to 2020, and the Associate Editor of the IEEE JOURNAL OF SELECTED AREAS IN COMMUNICATIONS 2020 in the area of distributed machine learning/federated learning.

Minh N. H. Nguyen (Member, IEEE) received the B.E. degree in computer science and engineering from Ho Chi Minh City University of Technology, Vietnam, in 2013, and the Ph.D. degree in computer science and engineering from Kyung Hee University, Seoul, South Korea, in 2020. He is currently working as a Lecturer at The University of Danang, Vietnam, and held a post-doctoral position with the Intelligent Networking Laboratory, Kyung Hee University, South Korea. His research interests include wireless communications, mobile edge computing, and federated learning.

Choong Seon Hong (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Kyung Hee University, Seoul, South Korea, in 1983 and 1985, respectively, and the Ph.D. degree from Keio University, Japan, in 1997. In 1988, he joined KT, South Korea, where he was involved in broadband networks as a member of the Technical Staff. Since 1993, he has been with Keio University. He was with the Telecommunications Network Laboratory, KT, as a Senior Member of Technical Staff and as the Director of the Networking Research Team until 1999. Since 1999, he has been a Professor with the Department of Computer Science and Engineering, Kyung Hee University. His research interests include future Internet, intelligent edge computing, network management, and network security.

Wei Bao (Member, IEEE) received the B.E. degree in communications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009, the M.A.Sc. degree in electrical and computer engineering from The University of British Columbia, Vancouver, Canada, in 2011, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, Canada, in 2016. He is currently a Senior Lecturer with the School of Computer Science, The University of Sydney, Sydney, Australia. His research covers the area of network science, with a particular emphasis on the Internet of Things, mobile computing, and edge computing. He received the Best Paper Awards in the ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM) in 2013 and 2019 and the IEEE International Symposium on Network Computing and Applications (NCA) in 2016.

Albert Y. Zomaya (Fellow, IEEE) is currently the Chair Professor of High-Performance Computing and Networking with the School of Computer Science, University of Sydney. He is also the Director of the Centre for Distributed and High-Performance Computing. He has published more than 700 scientific articles and is the author, coauthor, or editor of more than 25 books. He has delivered more than 190 keynote addresses, invited seminars, and media briefings and has been actively involved, in a variety of capacities, in the organization of more than 700 conferences. His research interests lie in parallel and distributed computing, networking, and complex systems.

Prof. Zomaya is a recipient of many awards, such as the IEEE Computer Society Technical Achievement Award (2014), the ACM MSWIM Reginald A. Fessenden Award (2017), and the New South Wales Premier's Prize of Excellence in Engineering and Information and Communications Technology (2019). He is a Chartered Engineer, a Fellow of the AAAS, IET (U.K.), an Elected Member of Academia Europaea, and an IEEE Computer Society's Golden Core member. He is the Founding Editor in Chief of the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING and the Editor-in-Chief of the ACM Computing Surveys. He previously served as the Editor-in-Chief for the IEEE TRANSACTIONS ON COMPUTERS (2011–2014). He currently serves as an Associate Editor for several leading journals.

Vincent Gramoli (Member, IEEE) is currently an invited Professor at the Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland. He is an Associate Professor of Distributed Computing at the University of Sydney, where he leads the Concurrent Systems Research Group. His research interest is in distributed computing, ranging from networking to blockchains. He has been affiliated with Université de Neuchâtel, INRIA, Cornell, and CSIRO in the past. He received the Future Fellowship of the Australian Research Council.