# Over-the-Air Federated Learning via Second-Order Optimization

Peng Yang, *Student Member, IEEE*, Yuning Jiang, *Member, IEEE*,
Ting Wang, *Senior Member, IEEE*, Yong Zhou, *Member, IEEE*,
Yuanming Shi, *Senior Member*, *IEEE*, Colin N. Jones, *Member, IEEE*

***Abstract*—Federated learning (FL) is a promising learning paradigm that can tackle the increasingly prominent isolated data islands problem while keeping users' data locally with privacy and security guarantees. However, FL could result in task-oriented data traffic flows over wireless networks with limited radio resources. To design communication-efficient FL, most of the existing studies employ the first-order federated optimization approach that has a slow convergence rate. This however results in excessive communication rounds for local model updates between the edge devices and edge server. To address this issue, in this paper, we instead propose a novel over-the-air second-order federated optimization algorithm to simultaneously reduce the communication rounds and enable low-latency global model aggregation. This is achieved by exploiting the waveform superposition property of a multi-access channel to implement the distributed second-order optimization algorithm over wireless networks. The convergence behavior of the proposed algorithm is further characterized, which reveals a linear-quadratic convergence rate with an accumulative error term in each iteration. We thus propose a system optimization approach to minimize the accumulated error gap by joint device selection and beamforming design. Numerical results demonstrate the system and communication efficiency compared with the state-of-the-art approaches.**

***Index Terms*—Federated learning, over-the-air computation, second-order optimization method**

## I. Introduction

Artificial intelligence (AI) technologies under rapid development have been widely studied and deployed in various scenarios. As a data-driven technology, its reliability and accuracy largely depend on the volume and quality of source data. However, it is recognized as a big challenge for most enterprises to obtain a dataset with sufficient volume and quality for AI model training. In the meantime, data privacy is another crucial issue that needs to be considered among different involved parties [1]. To this end, it is preferred in real-world implementations that data be kept locally, forming a variety of isolated data islands. This makes it difficult to directly aggregate data in the cloud and centrally train the AI models. Therefore, federated learning (FL) [2]–[4] has emerged as a novel paradigm to address these challenges. A generic and practical FL framework is essentially a distributed training process, and each iteration of FL includes the following three steps [4]. Firstly, the server broadcasts the current global model parameters to all the involved devices. Next, each device performs local model training based on its local data and then sends the local updates back to the server. Finally, the server aggregates the local updates and generates new global model parameters for the next iteration of distributed training. In essence, the server and devices aim to collaboratively solve a distributed optimization problem, which is typically referred to as *Federated Optimization* [5]. Different from centralized optimization, federated optimization confronts several practical challenges including communication efficiency, data heterogeneity, security, system complexity, etc. [6]. Among them, communication efficiency is of utmost importance since the communication between the server and devices usually suffers from unreliable network connections, limited resources, and severe latency [7].

To deal with the communication issue, a large amount of research has been conducted in federated optimization. On the one hand, reducing the communication volume in each iteration is an effective method. Specifically, quantization and sparsification techniques are employed to reduce the transmitted bits and remove the redundant updates of parameters, respectively [8], [9]. These compression techniques have shown remarkable effectiveness for high-dimensional models. However, their design needs to consider the compatibility for the aggregation operation in FL [6]. On the other hand, minimizing the total communication rounds is another primary method. To this end, zeroth-order methods [10], [11] have been investigated for some restrictive circumstances (e.g., black-box adversarial attack, non-smooth objective function) while showing great potential as only the objective function value is required to approximate derivative information [12]. In the situation where gradients are available, first-order methods are widely used. By increasing the amount of local computation, various gradient descent based methods have been shown that can significantly decrease the total number of communication rounds [2], [13], [14]. Nevertheless, these existing approaches, i.e., zeroth-order and first-order approaches, are

This work was supported in part by the Dean's Fund of Engineering Research Center of Software/Hardware Co-design Technology and Application, Ministry of Education (East China Normal University), the Natural Science Foundation of Shanghai under Grant 21ZR1442700, and the Swiss National Science Foundation under the RISK project (Risk Aware Data Driven Demand Response, grant number 200021 175627).

P. Yang and T. Wang are with the Shanghai Key Lab. of Trustworthy Computing; the Engineering Research Center of Software/Hardware Co-design Technology and Application, Ministry of Education; East China Normal University, Shanghai 200062, China (e-mail: 51205902030@stu.ecnu.edu.cn, twang@sei.ecnu.edu.cn).

Y. Jiang and C. N. Jones are with the Automatic Control Laboratory, EPFL, Laussane 1015, Switzerland (e-mail: yuning.jiang, colin.jones@epfl.ch).

Y. Zhou and Y. Shi are with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China (e-mail: zhouyong, shiym@shanghaitech.edu.cn).

governed by the linear convergence in the best case. As a result, the total number of iteration rounds required to achieve the desired accuracy is relatively large [15]. Therefore, the second-order methods (e.g., Newton-type methods) become attractive in such a wireless environment due to their fast local quadratic convergence rate. Nevertheless, the construction of the canonical Newton update requires both the Hessian and gradient information, where the distributed situation in FL makes gathering Hessian information a severe communication overhead. To this end, second-order federated optimization algorithms have been investigated to resolve this issue, which can be divided into two categories [15]. One is to use second-order information implicitly. In [16], a mirror descent update is carried out on the local function to approximate the Hessian information. In [17], the dual problems of the objective function are used to serve as the local subproblems. The other category is to use second-order information explicitly. In [18], a globally improved approximate Newton method (GIANT) using local Hessian for aggregation is proposed. In [19], [20], the optimization of the gradient's norm acts as the surrogate function. In [21], Hessian-vector product computation and conjugate gradient descent are performed on the devices and the server, respectively. The fast convergence rate with efficient communication makes the application of these second-order algorithms a great benefit to FL.

Despite the potential in the application of second-order algorithms to reduce the total communication rounds and improve the communication efficiency, the transmission of FL model parameters through wireless channels still confronts great challenges as wireless channels are always noisy with limited resources and high latency [22]–[24]. Based on the conventional "transmit-then-compute" principle, the aggregation of FL model parameters can be achieved by digital coded transmission and orthogonal multiple access (OMA) schemes [25]–[27]. By taking advantage of OMA and error correction techniques, local updates are transmitted separately in the quantized form and then decoded individually at the server. In this way, the model transmission can be deemed to be reliable and trustworthy. However, the increase in the number of devices will inevitably lead to a sharp increase in total communication latency and bandwidth requirement, which is often intolerable. Therefore, a novel technique called over-the-air computation (AirComp) [28] has emerged in FL algorithm design to decrease the communication cost based on the "compute-when-transmit" principle [25], [26], [29]–[39]. This technique leverages the superposition property of multiple access channels to realize the aggregation operation. Through the simultaneous transmission of all local updates, which are aggregated over the air, the communication overheads are significantly decreased. Specifically, the authors in [29] proposed an AirComp-based approach for FL with joint design of device selection and beamforming to improve the statistical learning performance. In [31], a novel Gradient-Based Multiple Access (GBMA) algorithm was put forward to perform FL with an energy scaling law for approaching the convergence rate of centralized training. In [34], the authors investigated the power control optimization for enhancing the learning performance of over-the-air federated learning. In [33], [40], intelligent reflecting surface (IRS) technology was used to achieve fast yet reliable model aggregation for over-the-air federated learning. The authors in [35] proposed the dynamic learning rate design for AirComp-based FL. Overall, the application of over-the-air computation in FL also improves the communication efficiency a lot.

Based on the above observations, this paper proposes to improve communication efficiency from two aspects, i.e., reducing communication rounds and the communication overhead in each round. To reduce the communication rounds, we shall utilize second-order information during the training process of FL. Due to the fast convergence speed, all these existing second-order state-of-the-arts have shown substantial improvement in terms of the total iteration rounds compared with first-order methods. However, their iterative procedures still have at least two communication rounds per iteration, i.e., the aggregation of gradient and second-order information. To avoid such two communication rounds, a recently proposed second-order method [41] cuts down the aggregation of gradients and realizes one communication round per iteration. Motivated by this, we adopt local Newton step aggregation for wireless FL algorithm design. Specifically, the product of the local Hessian's inversion and the local gradient is used to construct a local Newton step for aggregation. By this means, the devices only need to communicate once with the server per iteration, cutting down the transmission of local Hessian matrices and local gradients while keeping the convergence behavior of canonical Newton's method. Moreover, due to the limited radio resources, we adopt over-the-air computation, which has been widely used in the existing wireless FL schemes, to further reduce the communication overheads in each round. Based on this efficient local Newton step aggregation and AirComp technique, we propose an over-the-air second-order federated algorithm over wireless networks. Furthermore, we provide a rigorous theoretical analysis of the convergence behavior of our proposed method. The results show that the transmission of the above-mentioned product is sufficient to guarantee convergence and our proposed method outperforms first-order algorithms. To be specific, the proposed algorithm keeps a linear-quadratic convergence rate, which means it can achieve the optimal point with a quadratic convergence rate and degenerate into the linear convergence rate when it is close enough to the optimal point. However, as a result of local Newton step aggregation, device selection, and channel noise, there is an error term in each iteration. As the training proceeds, this accumulative error term will deflect the model parameters and affect learning performance. In order to mitigate the impact of this error term, we further propose a joint optimization approach of device selection and receiver beamforming. Specifically, Gibbs Sampling [42] is adopted to determine the set of selected devices, and the difference-of-convex-functions (DC) algorithm [43] is tailored to optimize the receiver beamforming during the iterative process of Gibbs Sampling.

### *A. Contributions*

In this paper, we propose a novel over-the-air FL algorithm via the second-order optimization method. Then, we theoreti-

cally analyze its convergence behavior, which shows that the proposed algorithm keeps a linear-quadratic convergence rate, with an accumulative error term arising during the FL process. To minimize the error gap and achieve better performance, we formulate this problem as a combinatorial non-convex problem and propose a system optimization approach to solve it. The main contributions of this paper are summarized as follows:

1) We design a novel AirComp-based FL algorithm by leveraging the principles of distributed second-order optimization methods and exploiting the waveform superposition property of a wireless multi-access channel for model aggregation. This algorithm is fundamentally different from most existing works which only consider gradient descent/SGD in training. The utilization of second-order information significantly reduces the total communication rounds in Aircomp-based FL, which further improves the communication efficiency.
2) We theoretically analyze the convergence behaviors of our proposed over-the-air second-order federated optimization algorithm with the presence of data heterogeneity (i.e., the different data sizes), device selection, and channel noise. The results show that our algorithm keeps a linear-quadratic convergence rate and outperforms first-order methods;
3) We formulate a system optimization problem to minimize the accumulative error gap during the execution of our proposed algorithm. Correspondingly, we propose a system optimization approach. Through the combination of Gibbs Sampling and DC algorithm, we jointly optimize the device selection and receiver beamforming;
4) We conduct extensive experiments to demonstrate that our proposed algorithm and system optimization approach can achieve better performance than other state-of-art approaches.

### B. Organization and Notations

The remainder of this paper is organized as follows. Section II presents the federated learning model and our FL algorithm. Section III provides the convergence analysis of our proposed algorithm. Section IV analyzes the system optimization problem arising from the error term, and describes our joint optimization method of device selection and beamforming. The experimental results are given in Section V. Finally, Section VI concludes the whole paper.

$\|\cdot\|_p$ is the $\ell_p$-norm, $\|\cdot\|_{\mathsf{F}}$ is the Frobenius norm. Italic, boldface lower case and upper case letters represent scalars, vectors and matrices, respectively. For a given set $\mathcal{X}$, $|\mathcal{X}|$ denotes the cardinality of $\mathcal{X}$. The operators $(\cdot)^{\mathsf{T}}$, $(\cdot)^{\mathsf{H}}$, $\mathrm{Tr}(\cdot)$ and $\mathrm{diag}(\cdot)$ denote the transpose, Hermitian transpose, trace, and diagonal matrix, respectively. $\mathbb{E}[\cdot]$ denotes the statistical expectation.

## II. Federated Learning Model and Algorithm

### A. Federated Learning System

A typical wireless federated learning system consists of a group of distributed devices and one server, where the communication takes place over wireless channels. As depicted in Fig. 1, there are $m$ single-antenna devices and a server equipped with $k$ antennas to collaboratively complete a learning task. We denote $\mathcal{D}$ as the entire sample set used in the FL task. Each device $i \in \mathcal{S}$ stores a sample set $\mathcal{D}_i = \{\boldsymbol{z}_{i,j} := (\boldsymbol{u}_{i,j}, v_{i,j})\}$ and $\mathcal{D} = \bigcup_{i=1}^{m} \mathcal{D}_i$ with $|\mathcal{D}| = n$, where $\mathcal{S}$ denotes the index set of devices, $\boldsymbol{u}_{i,j}$ is the feature vector and $v_{i,j}$ is the corresponding label.

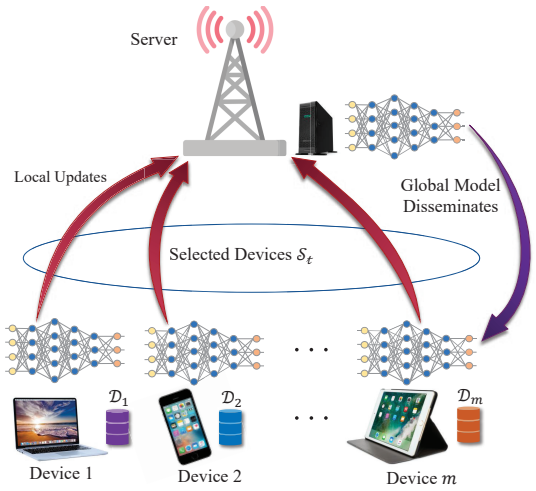



Fig. 1: Illustration of wireless FL systems.

As an important part of the learning task, the loss function is usually used for model parameter estimation. Here, the loss function of the $i$-th device is defined by

$$F_i(\boldsymbol{w}) = \frac{1}{|\mathcal{D}_i|} \sum_{\boldsymbol{z}_{i,j} \in \mathcal{D}_i} f(\boldsymbol{w}, \boldsymbol{z}_{i,j}) + \frac{\gamma}{2}\|\boldsymbol{w}\|_2^2 . \tag{1}$$

The first term is the average of $f(\boldsymbol{w}, \boldsymbol{z}_{i,j})$, where $\boldsymbol{w} \in \mathbb{R}^d$ is the model parameter vector and function $f$ is used to measure the prediction error of $\boldsymbol{w}$. The second term is for regularization with $\gamma$ being the weighting parameter. FL aims to train a suitable model at the server by aggregating the results collected from multiple devices, on which the distributed models are trained based on local datasets. Specifically, the server needs to optimize the following global loss function:

$$F(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{m} |\mathcal{D}_i| F_i(\boldsymbol{w}) . \tag{2}$$

### B. Federated Second-Order Optimization Algorithm

As typical training algorithms, gradient descent methods (e.g., SGD [44], batch gradient descent) are widely used. However, the relatively slow convergence rate of gradient descent results in too many communication rounds between the server and devices to complete the learning task. Thus, many research works have been done to improve the communication efficiency of gradient descent in FL. For example, some methods utilize multiple local updates to reduce the number of communication rounds [2], [13], while several algorithms employ compression techniques to reduce transmitted bits and save communication costs [8], [9], [45]. Although these schemes have greatly improved the communication efficiency

of gradient descent in FL, they are still limited by the linear convergence rate.

To address this issue, this paper considers second-order algorithms with a faster convergence rate such that the communication rounds can be significantly reduced. The descent direction vector of canonical Newton's method [46] is given by

$$\boldsymbol{p} = \left(\nabla^2 F\left(\boldsymbol{w}\right)\right)^{-1} \nabla F\left(\boldsymbol{w}\right). \tag{3}$$

The canonical Newton's method can achieve a locally quadratic convergence rate so that its total iteration rounds needed to complete the learning task are much fewer than first-order algorithms. However, in the distributed scenario, the computation of $\nabla^2 F\left(\boldsymbol{w}\right) = \frac{1}{m}\sum_{i=1}^{m} \nabla^2 F_i\left(\boldsymbol{w}\right)$ requires the aggregation of the local Hessian $\nabla^2 F_i\left(\boldsymbol{w}\right)$. The transmission of such $d \times d$ matrices inevitably brings huge communication overheads. To resolve this issue, numerous second-order distributed machine learning algorithms have been proposed, such as DANE [16], DISCO [21], GIANT [18], DINGO [19], and DINO [20]. These methods approximate Hessian information in varied forms to avoid the direct transmission of Hessian matrices and approach the performance of canonical Newton's method. However, at least two communication rounds per iteration are required, including the aggregation of local gradients and second-order descent directions. Different from these second-order algorithms, which require the aggregation of local gradients $\nabla F_i\left(\boldsymbol{w}\right)$ to compute the global gradient $\nabla F\left(\boldsymbol{w}\right) = \frac{1}{m}\sum_{i=1}^{m} \nabla F_i\left(\boldsymbol{w}\right)$, a recently proposed COMRADE [41] method cuts down this aggregation. By this means, the number of communication rounds required per iteration is reduced to one, further improving the communication efficiency. Motivated by this, we leverage the local Newton step aggregation as in [41] to achieve a faster convergence rate with fewer communication rounds. The product of the inversion of the local Hessian matrix $\left(\nabla^2 F_i\left(\boldsymbol{w}\right)\right)^{-1}$ and the local gradient $\nabla F_i\left(\boldsymbol{w}\right)$ is used to serve as the local descent direction vector $\boldsymbol{p}_i = \left(\nabla^2 F_i\left(\boldsymbol{w}\right)\right)^{-1} \nabla F_i\left(\boldsymbol{w}\right)$ for model aggregation. In this way, with the preserved convergence behavior of Newton's method, only one aggregation of the $d$-dimensional local descent direction vectors will be carried out in each iteration. To be specific, at $t$-th iteration, the procedure of our proposed method is summarized as follows:

1) **Device Selection**: The server decides the set of devices, denoted as $\mathcal{S}_t$, to participate in this iteration.
2) **Global Model Broadcast**: The server disseminates the current global model parameter vector $\boldsymbol{w}_t$ to the selected devices through the wireless channel.
3) **Local Model Update**: After the $i$-th device receives global model parameter vector $\boldsymbol{w}_t$, it first computes the local gradient based on local data samples:

$$\boldsymbol{g}_{t,i} = \nabla F_i\left(\boldsymbol{w}_t\right) = \frac{1}{|\mathcal{D}_i|}\sum_{\boldsymbol{z}_{i,j}\in\mathcal{D}_i} \nabla f\left(\boldsymbol{w}_t, \boldsymbol{z}_{i,j}\right) + \gamma \boldsymbol{w}_t \ , \tag{4}$$

where the derivatives are taken with respect to the first argument. Afterwards, the $i$-th device calculates the local Hessian matrix according to local gradient and local data samples:

$$\boldsymbol{H}_{t,i} = \nabla^2 F_i\left(\boldsymbol{w}_t\right) = \frac{1}{|\mathcal{D}_i|}\sum_{\boldsymbol{z}_{i,j}\in\mathcal{D}_i} \nabla^2 f\left(\boldsymbol{w}_t, \boldsymbol{z}_{i,j}\right) + \gamma \boldsymbol{I}_d \ . \tag{5}$$

The $i$-th device then gets a local Newton descent direction vector from previous results:

$$\boldsymbol{p}_{t,i} = \boldsymbol{H}_{t,i}^{-1}\boldsymbol{g}_{t,i} = \left(\nabla^2 F_i\left(\boldsymbol{w}_t\right)\right)^{-1} \nabla F_i\left(\boldsymbol{w}_t\right). \tag{6}$$

In practice, this step involves the computation of Hessian matrix and its inverse operation. To reduce the computational complexity, we adopt the conjugate gradient method [46] to obtain an approximate local Newton descent direction vector. According to the analysis in [18], this approximate solution will not have a significant impact on the convergence behavior.
4) **Model Aggregation**: The devices participating in the $t$-th iteration transmit local Newton descent direction vectors $\{\boldsymbol{p}_{t,i}\}$ to the server through the wireless channel, and the server aggregates them to obtain the global descent direction vector for this iteration:

$$\tilde{\boldsymbol{p}}_t = \frac{1}{\sum\limits_{i\in\mathcal{S}_t}|\mathcal{D}_i|}\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|\,\boldsymbol{p}_{t,i} \ . \tag{7}$$

5) **Global Model Update**: Finally, the server updates the model parameter vector $\boldsymbol{w}_t$ through global descent direction vector $\tilde{\boldsymbol{p}}_t$ and learning rate $\alpha$.

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha\tilde{\boldsymbol{p}}_t \ . \tag{8}$$

Notably, the Newton's method has a faster convergence rate than the gradient descent methods because it makes full use of the curvature information of the loss function, but the aggregation of the $d \times d$ local Hessian matrices for Newton descent direction in (3) aggravates the communication overheads in another way. As implied in Step 3) of our proposed FL scheme, it does not need to compute the global gradient $\nabla F\left(\boldsymbol{w}\right)$ and Hessian $\nabla^2 F\left(\boldsymbol{w}\right)$ to get a precise Newton descent direction by aggregating the local $\nabla F_i\left(\boldsymbol{w}\right)$ and $\nabla^2 F_i\left(\boldsymbol{w}\right)$. Note that this approximation also brings a controllable error gap with the exact descent direction vector, and its impact on the convergence rate will be analyzed in Section III.

### *C. Communication Model*

To further reduce the communication overheads, this subsection focuses on the design of the communication model between the server and devices. Specifically, there are two communication-related steps in each iteration of our FL algorithm. One is global model broadcasting in the downlink. Since only one global parameter vector needs to be broadcasted, the total communication cost of this step is negligible [25], [27], [31], [47]. The other is model aggregation in the uplink, which involves the transmission of $|\mathcal{S}_t|$ local descent direction vectors. Accordingly, the uploading process of this step brings the primary communication overhead in FL, which is also the focus of our communication model design.

In this paper, we consider a block fading channel. Each block is divided into $d$ time slots, ensuring the transmission

of one local descent direction vector. Suppose the traditional orthogonal multiple access channel is used to perform the model aggregation procedure. Each device will use one coherent block to transmit its local descent direction vector. Consequently, the time consumed for transmission in this step will increase linearly with the number of participating devices $|\mathcal{S}_t|$. Unfortunately, the number of devices $|\mathcal{S}_t|$ is usually very large, which inevitably leads to unacceptable communication overheads. In order to eliminate this issue, we adopt a state-of-the-art technique named over-the-air computation (AirComp) [28], which is shown to be effective in assisting the analog aggregation in FL studies [25], [26], [29]–[34]. This technique captures the nomographic function form of averaging the local descent direction vectors and implements the summation operation by the superposition property of the wireless channel. In this way, the server can receive the summation by letting all devices transmit their local descent direction vectors simultaneously in each block. Therefore, the entire process of model aggregation can be completed over the air in a single coherent block, and the communication overheads can be significantly reduced. More specifically, in the $t$-th iteration, the over-the-air computation can be represented as the nomographic function form [48] : $\hat{\boldsymbol{p}}_t = \psi\left(\sum_{i\in\mathcal{S}_t}\varphi_i\left(\boldsymbol{p}_{t,i}\right)\right)$. To reduce the transmission power, the pre-processing function $\phi_i$ and post-processing function $\psi$ can be designed to normalize and de-normalize the local descent direction vector $\boldsymbol{p}_{t,i}$ [30]. However, due to the variety of $\boldsymbol{p}_{t,i}$ among devices, the stationary of the information-bearing symbols obtained by such normalization methods can not be guaranteed, which further leads to the inapplicability of the uniform-forcing transceiver design in the following. Therefore, to guarantee the stationary of the information-bearing symbols, we adopt the data-and-CSI-aware design as in [49]. Before transmission, $\boldsymbol{p}_{t,i}$ is first pre-processed and encoded as $\boldsymbol{s}_{t,i}\in\mathbb{R}^d$ at the $i$-th device:

$$\boldsymbol{s}_{t,i} = \phi_i\left(\boldsymbol{p}_{t,i}\right) = \frac{|\mathcal{D}_i|\,\boldsymbol{p}_{t,i}}{\bar{p}_{t,i}}\ , \tag{9}$$

where $\bar{p}_{t,i} = |\mathcal{D}_i|\left\|\boldsymbol{p}_{t,i}\right\|_2$ is the product of the size of local dataset and the magnitude of $\boldsymbol{p}_{t,i}$. In this way, the stationary of the information-bearing symbols $\{\boldsymbol{s}_{t,i}\}$ can be guaranteed. Hence, we have $\left\|\boldsymbol{s}_{t,i}\right\|_2^2 = 1$ and $\mathbb{E}\left(|\boldsymbol{s}_{t,i}[j]|^2\right) = \frac{1}{d},\ \forall j\in d$, where $\boldsymbol{s}_{t,i}[j]$ denotes the $j$-th entry of $\boldsymbol{s}_{t,i}$. Thereafter, each entry of the transmitted signal sent by the $i$-th device is given by:

$$\boldsymbol{x}_{t,i}[j] = b_{t,i}\boldsymbol{s}_{t,i}[j]\ , \tag{10}$$

where $\boldsymbol{x}_{t,i}[j]\in\mathbb{R}$ and $\boldsymbol{s}_{t,i}[j]\in\mathbb{R}$ denote two representative entries of $\boldsymbol{x}_{t,i}$ and $\boldsymbol{s}_{t,i}$, respectively. $b_{t,i}\in\mathbb{R}$ is the transmitted power control factor, and the power constraint for each device in the whole process is given by:

$$\mathbb{E}\left(|b_{t,i}\boldsymbol{s}_{t,i}[j]|^2\right) = b_{t,i}^2/d \le P_0,\ \forall t,i\ , \tag{11}$$

where $P_0$ denotes the maximum transmitted power of each device.

Let $\boldsymbol{h}_{t,i}\in\mathbb{C}^k$ be the channel coefficient vector between the $i$-th device and the server in the $t$-th block, which remains unchanged in each block but differs among blocks. In addition, we assume that perfect channel state information (CSI) is available at all devices to adjust their transmitted signals based on channel coefficients [25], [29]–[32], [40], [50]–[52]. Then the received signal $\boldsymbol{y}_t\in\mathbb{C}^k$ at the server can be represented as follows:

$$\boldsymbol{y}_t = \sum_{i\in\mathcal{S}_t}\boldsymbol{h}_{t,i}\boldsymbol{x}_{t,i}[j] + \boldsymbol{e}_t = \sum_{i\in\mathcal{S}_t}\tilde{\boldsymbol{h}}_{t,i}b_{t,i}\,|\mathcal{D}_i|\,\boldsymbol{p}_{t,i}[j] + \boldsymbol{e}_t\ , \tag{12}$$

where $\tilde{\boldsymbol{h}}_{t,i} = \frac{\boldsymbol{h}_{t,i}}{\bar{p}_{t,i}}$ is the effective channel coefficient introduced in [49], $\boldsymbol{e}_t\in\mathbb{C}^k$ denotes the additive white Gaussian noise vector with the power of $\sigma^2$. We define the signal-to-noise ratio (SNR) as $P_0/\sigma^2$.

After the server receives $\boldsymbol{y}_t$, it can obtain the value $\boldsymbol{r}_t[j]\in\mathbb{C}$ before post-processing:

$$\begin{aligned}\boldsymbol{r}_t[j] &= \frac{1}{\sqrt{\eta_t}}\boldsymbol{a}_t^{\mathsf{H}}\boldsymbol{y}_t \\ &= \frac{1}{\sqrt{\eta_t}}\left(\boldsymbol{a}_t^{\mathsf{H}}\sum_{i\in\mathcal{S}_t}\tilde{\boldsymbol{h}}_{t,i}b_{t,i}\,|\mathcal{D}_i|\,\boldsymbol{p}_{t,i}[j] + \boldsymbol{a}_t^{\mathsf{H}}\boldsymbol{e}_t\right)\ ,\end{aligned} \tag{13}$$

where $\boldsymbol{a}_t\in\mathbb{C}^k$ represents the receiver beamforming vector and $\eta_t$ is the scaling factor. For convenience, we use $\boldsymbol{H}_t = [\tilde{\boldsymbol{h}}_{t,1},\ldots,\tilde{\boldsymbol{h}}_{t,|\mathcal{S}_t|}]$ to denote the effective channel coefficient matrix, $\boldsymbol{B}_t = \mathrm{diag}\left(b_{t,1},\ldots,b_{t,|\mathcal{S}_t|}\right)$ to denote the power transmission matrix, $\boldsymbol{G}_t = [|\mathcal{D}_1|\,\boldsymbol{p}_{t,1},\ldots,|\mathcal{D}_{|\mathcal{S}_t|}|\,\boldsymbol{p}_{t,|\mathcal{S}_t|}]^{\mathsf{T}}$ to denote the signal transmission matrix, and $\boldsymbol{E}_t = [\boldsymbol{e}_{t,1},\ldots,\boldsymbol{e}_{t,d}]$ to denote the noise matrix. So the total estimated value vector $\boldsymbol{r}_t = [\boldsymbol{r}_t[1],\ldots,\boldsymbol{r}_t[d]]$ can be written as:

$$\boldsymbol{r} = \frac{1}{\sqrt{\eta_t}}\left(\boldsymbol{a}^{\mathsf{H}}\boldsymbol{H}_t\boldsymbol{B}_t\boldsymbol{G}_t + \boldsymbol{a}_t^{\mathsf{H}}\boldsymbol{E}_t\right)\ . \tag{14}$$

To alleviate the influence of the distortion caused by noise and improve the performance of over-the-air computation, each entry of $\boldsymbol{B}_t$ follows the uniform-forcing transceiver design [53]:

$$b_{t,i} = \sqrt{\eta_t}\frac{\left(\boldsymbol{a}_t^{\mathsf{H}}\tilde{\boldsymbol{h}}_{t,i}\right)^{\mathsf{H}}}{\left\|\boldsymbol{a}_t^{\mathsf{H}}\tilde{\boldsymbol{h}}_{t,i}\right\|_2^2}\ . \tag{15}$$

where the transmission scalar $b_{t,i}$ can be computed after the calculation of receiver beamforming vector in system optimization, and then feed back to each device [29]. Substituting (15) into (14), we can get a simplified version of $\boldsymbol{r}_t$:

$$\boldsymbol{r}_t = \sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|\,\boldsymbol{p}_{t,i}^{\mathsf{T}} + \frac{1}{\sqrt{\eta_t}}\boldsymbol{a}_t^{\mathsf{H}}\boldsymbol{E}_t\ . \tag{16}$$

Finally, through the post-processing function of $\psi$, the server obtains the global descent direction vector $\hat{\boldsymbol{p}}_t$:

$$\begin{aligned}\hat{\boldsymbol{p}}_t = \psi\left(\boldsymbol{r}_t\right) &= \frac{1}{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}\boldsymbol{r}_t^{\mathsf{H}} \\ &= \tilde{\boldsymbol{p}}_t + \frac{1}{\left(\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|\right)\sqrt{\eta_t}}\left(\boldsymbol{a}_t^{\mathsf{H}}\boldsymbol{E}_t\right)^{\mathsf{H}}\ ,\end{aligned} \tag{17}$$

where $\tilde{\boldsymbol{p}}_{t,i}$ is the averaged local descent direction vector as defined in (7).

Based on the AirComp-based communication model and second-order optimization algorithm, we propose our over-the-air second-order federated algorithm, as shown in Algorithm 1.

**Algorithm 1:** Over-the-Air Second-Order Federated Algorithm

**for** *each iteration* $t$ **do**
- server chooses devices participating in this iteration and stores them as $\mathcal{S}_t$.
- server broadcasts the current model parameter vector $\boldsymbol{w}_t$ to all devices.
- **for** *each participating device* $i$ ***in parallel*** **do**
  - compute local gradient $\boldsymbol{g}_{t,i} = \frac{1}{|\mathcal{D}_i|}\sum_{\boldsymbol{z}_{i,j}\in\mathcal{D}_i}\nabla f\left(\boldsymbol{w}_t, \boldsymbol{z}_{i,j}\right) + \gamma \boldsymbol{w}_t$.
  - compute local Hessian matrix $\boldsymbol{H}_{t,i} = \frac{1}{|\mathcal{D}_i|}\sum_{\boldsymbol{z}_{i,j}\in\mathcal{D}_i}\nabla^2 f\left(\boldsymbol{w}_t, \boldsymbol{z}_{i,j}\right) + \gamma \boldsymbol{I}_d$.
  - compute local Newton descent direction $\boldsymbol{p}_{t,i} = \boldsymbol{H}_{t,i}^{-1}\boldsymbol{g}_{t,i}$.
  - encode $\boldsymbol{p}_{t,i}$ as $\boldsymbol{s}_{t,i}$ according to (9).
  - transmit the signal $\boldsymbol{x}_{t,i} = b_{t,i}\boldsymbol{s}_{t,i}$ through wireless channel.
- **end**
- server receives the signal $\boldsymbol{y}_t$ (12) and maintains $\hat{\boldsymbol{p}}_t$ (17).
- server performs an update step $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha\hat{\boldsymbol{p}}_t$.

**end**

## III. Theoretical Convergence Analysis

In this section, we provide the convergence analysis of our proposed algorithm. A major challenge of convergence analysis is to tackle the distortion of the descent direction vector caused by channel noise, device selection, and the use of local Newton step. To address this issue, we study the impact of distortion with respect to these influencing factors. In particular, we exploit the idea of sketching to analyze the approximation of local gradients and Hessian matrices. To better elaborate our analysis, some preliminaries are firstly presented.

### A. Preliminaries

The core of our proposed algorithm is using local Hessian matrices and local gradients, which is calculated through subsets of the total data set, to construct local Newton descent directions and aggregate them. This brings the benefits of fewer communication rounds between the server and the devices. However, since we rely on local information to approximate Newton descent directions, the quality of local Hessian/gradients, in other words, the difference between the local ones and global ones, are of concern. In order to tackle this issue, we adopt the idea of matrix sketching [54], [55]. Specifically, for a given input matrix $\boldsymbol{M} \in \mathbb{R}^{n\times d}$, we can replace it with $\boldsymbol{C} = \boldsymbol{L}^{\mathsf{T}}\boldsymbol{M} \in \mathbb{R}^{s\times d}$, where matrix $\boldsymbol{C}$ acts as the sketch of $\boldsymbol{M}$ with the sketching matrix $\boldsymbol{L} \in \mathbb{R}^{n\times s}$. In this way, the original problem related to $\boldsymbol{M}$ can be solved more efficiently using the smaller alternative matrix $\boldsymbol{C}$ without losing too much information. One may refer to [41, Appendix A] for more details of matrix sketching. The construction of the sketch is similar to the calculation of local Hessian/gradients, where we adopt partial information of the global Hessian/gradients to serve as the local Hessian/gradients. In this paper, we consider the row sampling scheme in matrix sketching. The sketch $\boldsymbol{C}$ is constructed by the uniform sampled and re-scaled subset of rows of $\boldsymbol{M}$ with sampling probability $\mathbb{P}\left(\boldsymbol{c}_i = \frac{\boldsymbol{m}_j}{\sqrt{sp}}\right) = p,\ p = \frac{1}{n}$, where $\boldsymbol{c}_i$ and $\boldsymbol{m}_j$ are the $i$-th row of $\boldsymbol{C}$ and $j$-th row of $\boldsymbol{M}$, respectively. Consequently, the sketching matrix $\boldsymbol{L}$ has only one non-zero entry in each column, and we shall measure the difference between the local Hessian/gradients and global ones with the help of such sketching matrices.

In the following, we consider a linear predictor model $\ell : \mathbb{R} \to \mathbb{R}$, which is frequently used in machine learning research, e.g., logistic and linear regression, support vector machines, neural networks and graphical models. The function $f\left(\boldsymbol{w}, \boldsymbol{z}_{i,j}\right)$ can thus be rewritten as $\ell\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{u}_{i,j}\right)$. Accordingly, we define $\boldsymbol{M}_t = \left[\boldsymbol{m}_1^{\mathsf{T}}, \ldots, \boldsymbol{m}_n^{\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^{n\times d}$ with $\boldsymbol{m}_{t,j} = \sqrt{\ell''\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{u}_{i,j}\right)/n}\boldsymbol{u}_{i,j} \in \mathbb{R}^d$, so the global Hessian matrix can be represented as $\boldsymbol{H}_t = \boldsymbol{M}_t^{\mathsf{T}}\boldsymbol{M}_t + \gamma\boldsymbol{I}_d$. Moreover, by defining $\boldsymbol{N}_t = [\boldsymbol{n}_1, \ldots, \boldsymbol{n}_n] \in \mathbb{R}^{d\times n}$ with $\boldsymbol{n}_i = \nabla f\left(\boldsymbol{w}_t, \boldsymbol{z}_i\right)$, the global gradient $\nabla F\left(\boldsymbol{w}_t\right)$ can be denoted by $\boldsymbol{g}_t = \frac{1}{n}\boldsymbol{N}_t\boldsymbol{1} + \gamma\boldsymbol{w}_t$. Let $\{\boldsymbol{L}_i\}_{i=1}^m$ be the sketching matrices, the local Hessian matrices and local gradients can be reformulated as:

$$\boldsymbol{H}_{t,i} = \boldsymbol{M}_t^{\mathsf{T}}\boldsymbol{L}_i\boldsymbol{L}_i^{\mathsf{T}}\boldsymbol{M}_t + \gamma\boldsymbol{I}_d\ , \quad \boldsymbol{g}_{t,i} = \frac{1}{n}\boldsymbol{N}_t\boldsymbol{L}_i\boldsymbol{L}_i^{\mathsf{T}}\boldsymbol{1} + \gamma\boldsymbol{w}\ , \tag{18}$$

which are approximations to the true global Hessians and gradients, and the analysis will characterize the error introduced by this approximation. In addition, we define an auxiliary quadratic function as follows to facilitate our analysis:

$$\phi(\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{H}_t\boldsymbol{p} - \boldsymbol{g}_t^{\mathsf{T}}\boldsymbol{p} = \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\left(\boldsymbol{M}_t^{\mathsf{T}}\boldsymbol{M}_t + \gamma\boldsymbol{I}_d\right)\boldsymbol{p} - \boldsymbol{g}_t^{\mathsf{T}}\boldsymbol{p}\ . \tag{19}$$

As a quadratic function, the minimum point of $\phi(\boldsymbol{p})$ denoted by $\boldsymbol{p}^*$ can be analytically obtained, which is the same as the exact Newton descent direction vector in (3), i.e.,

$$\begin{aligned}\boldsymbol{p}^* = \arg\min\phi(\boldsymbol{p}) &= \nabla^2 F^{-1}\left(\boldsymbol{w}_t\right)\nabla F\left(\boldsymbol{w}_t\right)\\ &= \boldsymbol{H}_t^{-1}\boldsymbol{g}_t = \left(\boldsymbol{M}_t^{\top}\boldsymbol{M}_t + \gamma\boldsymbol{I}_d\right)^{-1}\boldsymbol{g}_t\ .\end{aligned} \tag{20}$$

Due to the effect of channel noise, device selection, and the use of local Newton step, the actual descent direction rather than the exact Newton step $\boldsymbol{p}^*$ is given by:

$$\hat{\boldsymbol{p}}_t = \boldsymbol{p}^* + \underbrace{\left(\bar{\boldsymbol{p}}_t - \boldsymbol{p}^*\right)}_{\text{Local Hessian}} + \underbrace{\left(\boldsymbol{p}_t - \bar{\boldsymbol{p}}_t\right)}_{\text{Local Gradient}} + \underbrace{\left(\tilde{\boldsymbol{p}}_t - \boldsymbol{p}_t\right)}_{\text{Device Selection}} + \underbrace{\left(\hat{\boldsymbol{p}}_t - \tilde{\boldsymbol{p}}_t\right)}_{\text{Channel Noise}}\ , \tag{21}$$

where $\hat{\boldsymbol{p}}_t$ and $\tilde{\boldsymbol{p}}_t$ are defined in (17) and (7), respectively, $\boldsymbol{p}_t = \frac{1}{\sum_{i\in\mathcal{S}}|\mathcal{D}_i|}\sum_{i\in\mathcal{S}}|\mathcal{D}_i|\,\boldsymbol{p}_{t,i}$ is the averaged local descent direction vector without device selection, and $\bar{\boldsymbol{p}}_t = \frac{1}{\sum_{i\in\mathcal{S}}|\mathcal{D}_i|}\sum_{i\in\mathcal{S}}|\mathcal{D}_i|\,\bar{\boldsymbol{p}}_{t,i} = \frac{1}{\sum_{i\in\mathcal{S}}|\mathcal{D}_i|}\sum_{i\in\mathcal{S}}|\mathcal{D}_i|\,\boldsymbol{H}_{t,i}^{-1}\boldsymbol{g}_t$ is the Newton descent direction with the exact global gradient. It is obvious that device selection and noise play a significant role in the difference between $\hat{\boldsymbol{p}}_t$ and $\boldsymbol{p}^*$, and it is critical to investigate the impact of them on the convergence of the proposed algorithm. In the following analysis, we will use the quadratic function (19) to illustrate how close $\hat{\boldsymbol{p}}_t$ and $\boldsymbol{p}^*$ are. Besides, the error of model parameter vector in the iterates

$\boldsymbol{\Delta}_t = \boldsymbol{w}_t - \boldsymbol{w}^*$ acts as the metric, where $\boldsymbol{w}^*$ denotes the optimal solution.

Throughout this paper, we consider the following assumptions, which are widely adopted in FL problems [33], [34], [56].

**Assumption 1.** The global loss function $F$ is $L$-smooth.

**Assumption 2.** The global loss function $F$ is strongly convex, which indicates the unique optimal model parameter vector $\boldsymbol{w}^*$ of the FL task.

**Assumption 3.** The local loss function $F_i$ is twice-differentiable, smooth and convex.

### *B. Convergence Analysis*

Since the local gradients and Hessian matrices are adopted to approximate the global descent direction, the gap between the local direction and the global direction is essential for the convergence analysis. Therefore, we first recall two lemmas to reveal their relationships.

**Lemma 1** ( [41, variant of Lemma 2])**.** Let $\lambda, \delta = \sum_{i=1}^m \delta_i, \{\delta_i\} \in (0,1)$ be fixed parameters, $r = \operatorname{rank}(\boldsymbol{M}_t)$, and $\boldsymbol{U} \in \mathbb{R}^{n\times r}$ be the orthonormal bases of the matrix $\boldsymbol{M}_t$. Let $\mu \in \left[1, \frac{n}{d}\right]$ be the coherence of $\boldsymbol{M}_t$ defined in [30]. Let $\left\{\boldsymbol{L}_i \in \mathbb{R}^{n\times|\mathcal{D}_i|}\right\}_{i=1}^m$ be independent uniform sampling sketching matrices with $|\mathcal{D}_i| \geq \frac{3\mu d}{\lambda^2}\log\frac{d}{\delta_i}$. It holds with the probability exceeding $1-\delta$ that:

$$\left\|\boldsymbol{U}^\mathsf{T}\boldsymbol{L}_i\boldsymbol{L}_i^\mathsf{T}\boldsymbol{U} - \boldsymbol{I}\right\|_2 \leq \lambda\ , \quad \forall i \in \mathcal{S}\ . \tag{22}$$

**Lemma 2** ( [41, variant of Lemma 3])**.** Let $\left\{\boldsymbol{L}_i \in \mathbb{R}^{n\times|\mathcal{D}_i|}\right\}_{i=1}^m$ be independent uniform sampling sketching matrices, $\delta = \sum_{i=1}^m \delta_i, \{\delta_i\} \in (0,1)$ be fixed parameters, then with the probability exceeding $1-\delta$, we have:

$$\begin{aligned}&\left\|\frac{1}{n}\boldsymbol{N}_t\boldsymbol{L}_i\boldsymbol{L}_i^\mathsf{T}\mathbf{1} - \frac{1}{n}\boldsymbol{N}_t\mathbf{1}\right\| \\ \leq&\left(1+\sqrt{2\ln\left(\frac{1}{\delta_i}\right)}\right)\sqrt{\frac{1}{|\mathcal{D}_i|}}\max_j\|\boldsymbol{n}_j\|\ .\end{aligned} \tag{23}$$

With Lemma 1 and Lemma 2, we further propose Lemma 3 to characterize the gap between $\hat{\boldsymbol{p}}_t$ and $\boldsymbol{p}^*$ via the support quadratic function.

**Lemma 3.** Let $\{\boldsymbol{L}_i\}_{i=1}^m \in \mathbb{R}^{n\times|\mathcal{D}_i|}$ be independent uniform sampling sketching matrices, $\phi_t$ be the quadratic function as defined in (19), $\lambda, \{\delta_i\} \in (0,1)$ be fixed parameters with $\tilde{\delta} = \min\{\delta_i\}$ and $\hat{\boldsymbol{p}}_t$ be the approximate descent direction vector defined in (21). It holds that:

$$\phi_t(\boldsymbol{p}^*) \leq \phi_t(\hat{\boldsymbol{p}}_t) \leq \epsilon^2 + \left(1-\zeta^2\right)\phi_t(\boldsymbol{p}^*)\ ,$$

where

$$\zeta^2 = 3\tau^2\left(\lambda + \frac{\lambda^2}{1-\lambda}\right)^2 + 24\vartheta^2\left(\tau\left(\lambda+\frac{\lambda^2}{1-\lambda}\right)+1\right)^2 \tag{24}$$

with $\tau = \frac{\sigma_{\max}\left(\boldsymbol{M}^\top\boldsymbol{M}\right)}{\sigma_{\max}(\boldsymbol{M}^\top\boldsymbol{M})+n\gamma}$, $\vartheta = \max_t\left(1 - \frac{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}{n}\right) < 1$ and

$$\begin{aligned}\epsilon^2 =& \frac{3}{\sigma_{\min}(\boldsymbol{H}_t)}\left\|\frac{1}{\left(\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|\right)\sqrt{\eta_t}}\boldsymbol{a}_t^\mathsf{H}\boldsymbol{E}_t\right\|^2 \\ &+\left[24\left(1-\frac{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}{n}\right)^2\frac{1}{\min_{i\in\mathcal{S}_t}|\mathcal{D}_i|}+\frac{m}{n}\right] \\ &\cdot\left[\frac{1}{1-\lambda}\frac{1}{\sqrt{\sigma_{\min}(\boldsymbol{H}_t)}}\left(1+\sqrt{2\ln\left(\frac{1}{\tilde{\delta}}\right)}\right)\max_j\|\boldsymbol{n}_j\|\right]^2.\end{aligned} \tag{25}$$

The proof of Lemma 3 can be found in Appendix A. To illustrate that $\hat{\boldsymbol{p}}_t$ is a good descending direction, we introduce Lemma 4 supported by the property of the quadratic function introduced in Lemma 3.

**Lemma 4** ( [41, Lemma 6])**.** Let $\zeta \in (0,1)$, $\epsilon$ be any fixed parameter, if $\hat{\boldsymbol{p}}_t$ satisfies $\phi(\hat{\boldsymbol{p}}_t) \leq \epsilon^2 + (1-\zeta^2)\min_{\boldsymbol{p}}\phi(\boldsymbol{p})$, then under Assumption 1, the error of model parameter vector $\boldsymbol{\Delta}_t = \boldsymbol{w}_t - \boldsymbol{w}^*$ in iterations satisfies

$$\boldsymbol{\Delta}_{t+1}^\mathsf{T}\boldsymbol{H}_t\boldsymbol{\Delta}_{t+1} \leq L\|\boldsymbol{\Delta}_{t+1}\|\|\boldsymbol{\Delta}_t\|^2 + \frac{\zeta^2}{1-\zeta^2}\boldsymbol{\Delta}_t^\mathsf{T}\boldsymbol{H}_t\boldsymbol{\Delta}_t + 2\epsilon^2\ , \tag{26}$$

Based on Lemma 3 and Lemma 4, we can derive the main result:

**Theorem 1.** Suppose the size of local dataset at each device $|\mathcal{D}_i| \geq \frac{3\mu d}{\lambda^2}\log\frac{d}{\delta_i}$ for some $\lambda, \delta_i \in (0,1)$, then under Assumption 1 with the probability exceeding $1-\delta$ we have

$$\begin{aligned}&\mathbb{E}\left(\|\boldsymbol{\Delta}_{t+1}\|\right) \\ \leq&\max\left\{\sqrt{\kappa_t\left(\frac{\zeta^2}{1-\zeta^2}\right)}\|\boldsymbol{\Delta}_t\|, \frac{L}{\sigma_{\min}(\boldsymbol{H}_t)}\|\boldsymbol{\Delta}_t\|^2\right\} + \epsilon'\ ,\end{aligned}$$

where the expectation takes with respect to the channel noise $\boldsymbol{e}_t$, $\zeta$ is defined as (24), $\kappa_t = \frac{\sigma_{\max}(\boldsymbol{H}_t)}{\sigma_{\min}(\boldsymbol{H}_t)}$ denotes the condition number of $\boldsymbol{H}_t$, and

$$\begin{aligned}\epsilon' =& \frac{2\sqrt{3}}{\sigma_{\min}(\boldsymbol{H}_t)}\frac{d\sigma}{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}\frac{\|\boldsymbol{a}_t\|_2}{\sqrt{\eta_t}} \\ &+\sqrt{24\left(1-\frac{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}{n}\right)^2\frac{1}{\min_{i\in\mathcal{S}_t}|\mathcal{D}_i|}+\frac{m}{n}} \\ &\cdot\frac{1}{1-\lambda}\frac{2}{\sigma_{\min}(\boldsymbol{H}_t)}\left(1+\sqrt{2\ln\left(\frac{1}{\tilde{\delta}}\right)}\right)\max_j\|\boldsymbol{n}_j\|\ .\end{aligned}$$

The proof can be found in Appendix B. From Theorem 1, we have the following observations.

*1)* **The proposed algorithm keeps a linear-quadratic convergence rate:** From the analysis results, it can be seen that the term $\|\boldsymbol{\Delta}_t\| = \|\boldsymbol{w}_t - \boldsymbol{w}^*\|$ keeps the property in this form: $\mathbb{E}\left(\|\boldsymbol{\Delta}_{t+1}\|\right) \leq \max\left\{\omega_1\|\boldsymbol{\Delta}_t\|, \omega_2\|\boldsymbol{\Delta}_t\|^2\right\} + \epsilon'$. When $\|\boldsymbol{\Delta}_t\| > \frac{\omega_1}{\omega_2}$, this property can be simplified as $\mathbb{E}\left(\|\boldsymbol{\Delta}_{t+1}\|\right) \leq \omega_2\|\boldsymbol{\Delta}_t\|^2 + \epsilon'$. It is obvious that the proposed algorithm keeps the same quadratic convergence rate as the canonical Newton's method. At the beginning of the algorithm it can converge to the neighbor of the optimal point quickly. When $\|\boldsymbol{\Delta}_t\| < \frac{\omega_1}{\omega_2}$,

this property turns into $\mathbb{E}\left(\|\boldsymbol{\Delta}_{t+1}\|\right) \leq \omega_1 \|\boldsymbol{\Delta}_t\| + \epsilon'$, which means when $\|\boldsymbol{\Delta}_t\|$ is small enough during the process of the algorithm, it degenerates into the linear convergence rate. In conclusion, the proposed algorithm keeps a linear-quadratic convergence rate and performs better than first-order algorithms.

*2)* **The proposed algorithm is accompanied by an accumulative error term:** Notice that there is an error term $\epsilon'$ in each iteration, which comes from the approximation, device selection and channel noise. Consider the noise-free case without device selection, which means $\sigma = 0$ and $|\mathcal{S}_t| = m$, this error term degenerates to:

$$\epsilon_1' = \frac{1}{1-\lambda}\frac{2}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left(1+\sqrt{2\ln\left(\frac{1}{\tilde{\delta}}\right)}\right)\sqrt{\frac{m}{n}}\max_j \|\boldsymbol{n}_j\| \ , \tag{27}$$

which is exactly the same as the error term introduced in [41]. With the algorithm executed iteratively, the gap between the expected global loss function value and the optimal one is upper bounded by this accumulative error term. Therefore, the active device set $\mathcal{S}_t$, the receiver beamforming vectors $\{\boldsymbol{a}_t\}$ and the scaling factors $\{\eta_t\}$ need to be tuned in each iteration so as to reduce the error gap.

## IV. System Optimization

In this section, we first formulate a system optimization problem to minimize the error term in the convergence analysis results. Then, we propose our approach for joint optimization of device selection and receiver beamforming vector.

### A. Problem Formulation

In light of convergence analysis, to obtain a precise model parameter vector, minimizing the error gap demonstrated in Theorem 1 is a key issue. It is observed that the coefficients of the two iterative terms $\boldsymbol{\Delta}_t$ and $\boldsymbol{\Delta}_{t+1}$ in Theorem 1 are independent of variables $\mathcal{S}_t$, $\boldsymbol{a}_t$ and $\eta_t$. Therefore, in order to achieve the minimization of the total error gap, we only need to minimize the error term $\epsilon'$ in each iteration as follows

$$\begin{aligned}\min_{\mathcal{S}_t,\boldsymbol{a}_t,\eta_t} \quad & \frac{2\sqrt{3}}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\frac{d\sigma}{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}\frac{\|\boldsymbol{a}_t\|_2}{\sqrt{\eta_t}} \\ & + \sqrt{24\left(1-\frac{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}{n}\right)^2\frac{1}{\min_{i\in\mathcal{S}_t}|\mathcal{D}_i|}+\frac{m}{n}} \\ & \cdot\frac{1}{1-\lambda}\frac{2}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left(1+\sqrt{2\ln\left(\frac{1}{\tilde{\delta}}\right)}\right)\max_j\|\boldsymbol{n}_j\| \\ \text{s.t.} \quad & \frac{\eta_t}{\left\|\boldsymbol{a}_t^{\mathsf{H}}\tilde{\boldsymbol{h}}_{t,i}\right\|^2} \leq dP_0 \quad \forall i\in\mathcal{S}_t\end{aligned} \tag{28}$$

The power constraint in (28) can be rewritten in the form of the restriction of scaling factor: $\eta_t \leq dP_0\left\|\boldsymbol{a}_t^{\mathsf{H}}\tilde{\boldsymbol{h}}_{t,i}\right\|^2$, $\forall i\in\mathcal{S}_t$. We take the negative correlation between the scaling factor $\eta_t$ and the objective function value $\epsilon'$ into consideration. $\eta_t$ can be set as $\eta_t = dP_0\min_{i\in\mathcal{S}_t}\left\|\boldsymbol{a}_t^{\mathsf{H}}\tilde{\boldsymbol{h}}_{t,i}\right\|^2$ [53], and the problem can be simplified as $\mathscr{P}$:

$$\begin{aligned}\mathscr{P}: \min_{\mathcal{S}_t,\boldsymbol{a}_t} \ & \frac{\sqrt{3d}\sigma}{\sqrt{P_0}\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}\max_{i\in\mathcal{S}_t}\left(\frac{\|\boldsymbol{a}_t\|}{\left\|\boldsymbol{a}_t^{\mathsf{H}}\tilde{\boldsymbol{h}}_{t,i}\right\|}\right) \\ & + \sqrt{24\left(1-\frac{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}{n}\right)^2\frac{1}{\min_{i\in\mathcal{S}_t}|\mathcal{D}_i|}+\frac{m}{n}} \\ & \cdot\frac{1}{1-\lambda}\frac{2}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left(1+\sqrt{2\ln\left(\frac{1}{\tilde{\delta}}\right)}\right)\max_j\|\boldsymbol{n}_j\| \ .\end{aligned} \tag{29}$$

We have the following key observations for solving (29):

- Intuitively, to achieve the minimization of the objective value of $\mathscr{P}$ , the number of selected devices is supposed to be maximized, then $\mathscr{P}$ will degenerate into the form of traditional beamforming optimization. However, the term $\max_{i\in\mathcal{S}_t}\left(\frac{\|\boldsymbol{a}_t\|}{\left\|\boldsymbol{a}_t^{\mathsf{H}}\tilde{\boldsymbol{h}}_{t,i}\right\|}\right)$ is related to device selection, which further results in the incorrectness of maximizing $|\mathcal{S}_t|$ directly.
- By searching over all the possible participating device sets, the optimal $\mathcal{S}_t$ can be determined. Still, the number of devices $m$ can be very large, leading to an exponential growth of the optimization procedure in the number of devices $m$.
- After the search of participating devices, the remaining problem is a typical beamforming optimization problem, but it is still non-convex and intractable.

In conclusion, since a combinatorial search of participating devices and minimization of the non-convex objective function are involved, it is evident that $\mathscr{P}$ is a mixed-integer non-convex problem. In order to tackle the complexity of computation and the difficulty of non-convexity, we propose an efficient method to iteratively search the optimal set of selected devices $\mathcal{S}_t$ while jointly optimizing the receiver beamforming vector $\boldsymbol{a}_t$ for each given $\mathcal{S}_t$.

### B. Receiver Beamforming Optimization

For a given set of selected devices $\mathcal{S}_t$, $\mathscr{P}$ can be simplified as $\mathscr{P}_1 : \min_{\boldsymbol{a}_t}\max_{i\in\mathcal{S}_t}\frac{\|\boldsymbol{a}_t\|}{\left\|\boldsymbol{a}_t^{\mathsf{H}}\tilde{\boldsymbol{h}}_{t,i}\right\|}$, which is equivalent to: $\min_{\boldsymbol{a}_t}\max_{i\in\mathcal{S}_t}\frac{\|\boldsymbol{a}_t\|^2}{\left\|\boldsymbol{a}_t^{\mathsf{H}}\tilde{\boldsymbol{h}}_{t,i}\right\|^2}$ . This can be further reformulated as $\mathscr{P}_1'$ according to the analysis in [53]:

$$\mathscr{P}_1': \min_{\boldsymbol{a}_t} \quad \|\boldsymbol{a}_t\|^2 \qquad \text{s.t.} \quad \left\|\boldsymbol{a}_t^{\mathsf{H}}\tilde{\boldsymbol{h}}_{t,i}\right\|^2 \geq 1 \quad \forall i\in\mathcal{S}_t \ .$$

It can be seen that $\mathscr{P}_1'$ is actually a quadratically constrained quadratic programming problem, which is difficult to solve. We first use the matrix lifting technique to pre-process $\mathscr{P}_1'$ and turn it into a low-rank optimization form. Specifically, let $\boldsymbol{A} = \boldsymbol{a}_t\boldsymbol{a}_t^{\mathsf{H}}$ with $\text{rank}\left(\boldsymbol{A}\right) = 1$ and $\boldsymbol{Q}_i = \tilde{\boldsymbol{h}}_{t,i}\tilde{\boldsymbol{h}}_{t,i}^{\mathsf{H}}$, $\mathscr{P}_1'$ can be recast as:

$$\begin{aligned}\min_{\boldsymbol{A}} \quad & \text{Tr}\left(\boldsymbol{A}\right) \\ \text{s.t.} \quad & \boldsymbol{A}\succeq\boldsymbol{0}, \ \ \text{rank}(\boldsymbol{A}) = 1, \ \ \text{Tr}\left(\boldsymbol{A}\boldsymbol{Q}_i\right) \geq 1 \quad \forall i\in\mathcal{S}_t \ .\end{aligned}$$

The key to solving this low-rank optimization problem is to deal with the troublesome rank-one constraint. A common method to solve such a problem is semidefinite relaxation (SDR) [57], [58], which drops the rank-one constraint to obtain a relaxed problem in the form of semidefinite programming. By this means, SDR can arrive at an approximate solution efficiently through solving the relaxed problem. However, as the size of the problem grows, the rank-one constraint is usually unsatisfied. In this situation, the approximate solution needs to be scaled through randomization methods, leading to an alternative solution with low accuracy [53], which will further affect the learning performance of FL. To guarantee the rank-one constraint, we can replace it with its equivalent form [29], [59]: $\mathrm{Tr}\left(\boldsymbol{A}\right) - \left\|\boldsymbol{A}\right\|_2 = 0 \quad \text{with} \quad \mathrm{Tr}\left(\boldsymbol{A}\right) > 0$ . Then, the original problem turns into a difference-of-convex-function (DC) program. By solving this DC program, a more precise solution can be obtained since all constraints are satisfied. Therefore, we develop a DC Algorithm (DCA) based on the principles in [43], [60] to solve this problem. Specifically, we can get the following problem by taking the new constraint as a penalty term:

$$\begin{aligned} \min_{\boldsymbol{A}} \quad & \mathrm{Tr}\left(\boldsymbol{A}\right) + \theta\left(\mathrm{Tr}\left(\boldsymbol{A}\right) - \left\|\boldsymbol{A}\right\|_2\right) \\ \text{s.t.} \quad & \boldsymbol{A} \succeq \boldsymbol{0},\ \mathrm{Tr}(\boldsymbol{A}) > 0,\ \mathrm{Tr}\left(\boldsymbol{A}\boldsymbol{Q}_i\right) \geq 1 \ \ \forall i \in \mathcal{S}_t\ , \end{aligned}$$

where $\theta$ is the penalty factor. Although this is still a non-convex problem owing to the concave term $-\left\|\boldsymbol{A}\right\|_2$, we can take the linearization of $\left\|\boldsymbol{A}\right\|_2$ and convert it into a convex subproblem:

$$\begin{aligned} \mathscr{P}_{DCA}: \min_{\boldsymbol{A}} \ & (1+\theta)\,\mathrm{Tr}(\boldsymbol{A}) - \theta\left\langle \partial\left\|\boldsymbol{A}_j\right\|_2, \boldsymbol{A}\right\rangle \\ & \text{s.t. } \boldsymbol{A} \succeq \boldsymbol{0},\ \mathrm{Tr}(\boldsymbol{A}) > 0,\ \mathrm{Tr}\left(\boldsymbol{A}\boldsymbol{Q}_i\right) \geq 1\ \forall i \in \mathcal{S}_t, \end{aligned}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product of two matrices and $\partial\left\|\boldsymbol{A}_j\right\|_2$ represents the subgradient of $\left\|\boldsymbol{A}_j\right\|_2$ at $\boldsymbol{A}_j$. The overall procedure of DCA is as summarized in Algorithm 2. In addition, one can refer to [60] for the convergence guarantee of DCA.

**Algorithm 2:** DC Algorithm for Receiver Beamforming Optimization (DCA)

**input:** effective channel coefficients $\left\{\tilde{\boldsymbol{h}}_{t,i}\right\}$, penalty factor $\theta$, threshold $\xi$
turn $\mathscr{P}$ into the DCA form $\mathscr{P}_{DCA}$.
choose $\boldsymbol{A}_0 \succeq 0$, set $j = 1$.
**while** $\left|\mathrm{Tr}\left(\boldsymbol{A}_{j-1}\right) - \left\|\boldsymbol{A}_{j-1}\right\|_2\right| \geq \xi$ **do**
  compute the subgradient $\partial\left\|\boldsymbol{A}_{j-1}\right\|_2$.
  substitute $\partial\left\|\boldsymbol{A}_{j-1}\right\|_2$ into $\mathscr{P}_{DCA}$, solve the subproblem and set the result as $\boldsymbol{A}_j$.
  $j \leftarrow j+1$ .
**end**

### C. Device Selection Optimization

As mentioned above, the device selection is a combinatorial optimization problem, which is impossible to perform a traversal in the whole solution space. Thus, we adopt the well-known Gibbs Sampling (GS) [42] method to optimize the selection of device set iteratively. The main idea of GS is that in each iteration, a device set is sampled from the neighbors of the current device set according to an appropriate distribution. In this way, the set of selected devices can gradually approach the global optimal solution.

**Algorithm 3:** System optimization approach GS+DCA

**input:** effective channel coefficients $\left\{\tilde{\boldsymbol{h}}_{t,i}\right\}$, $T^{(0)}$, $\rho$, $K$
**output:** $\mathcal{S}^{(k+1)}$ and its corresponding $\boldsymbol{a}^{(K+1)}$.
**initialization:** $\mathcal{S}^{(0)} = \mathcal{S}$
**for** *iteration* $k = 0, 1, 2, ..., K$ **do**
  generate the neighboring solution set $\mathcal{F}^{(k)}$.
  **for** *each* $\tilde{\mathcal{S}} \in \mathcal{F}^{(k)}$ **do**
    substitute $\tilde{\mathcal{S}}$ into $\mathscr{P}_1$, then solve the problem using DCA to get the corresponding optimal $\tilde{\boldsymbol{a}}$.
  **end**
  sample $\tilde{\mathcal{S}}^{(k)}$ according to the probability $\mathbb{P}\left(\tilde{\mathcal{S}}^{(k)}\right) = \frac{\exp\left(-J\left(\tilde{\mathcal{S}}^{(k)}, \tilde{\boldsymbol{a}}^{(k)}\right)/T^{(k)}\right)}{\sum_{\tilde{\mathcal{S}}\in\mathcal{F}^{(k)}}\exp\left(-J\left(\tilde{\mathcal{S}}, \tilde{\boldsymbol{a}}\right)/T^{(k)}\right)}$.
  $\mathcal{S}^{(k+1)} \leftarrow \tilde{\mathcal{S}}^{(k)}$, $T^{(k+1)} \leftarrow \rho T^{(k)}$.
**end**

To be specific, we treat different sets of selected devices as states, and the goal is to find the state which can minimize the objective value in $\mathscr{P}$. For the sake of such state, at iteration $k$ of GS's process, with the set of selected devices $\mathcal{S}^{(k-1)}$ given in the last iteration, we first generate the neighboring solution set of $\mathcal{S}^{(k-1)}$. The neighboring solution set, denoted by $\mathcal{F}^{(k)}$, contains the device sets that differ from the $\mathcal{S}^{(k-1)}$ in only one entry. For example, by assuming $\mathcal{S} = \{0, 1, 2\}$ and $\mathcal{S}^{(k-1)} = \{1, 2\}$, then we have $\mathcal{F}^{(k)} = \{\{0, 1, 2\}, \{2\}, \{1\}\}$.

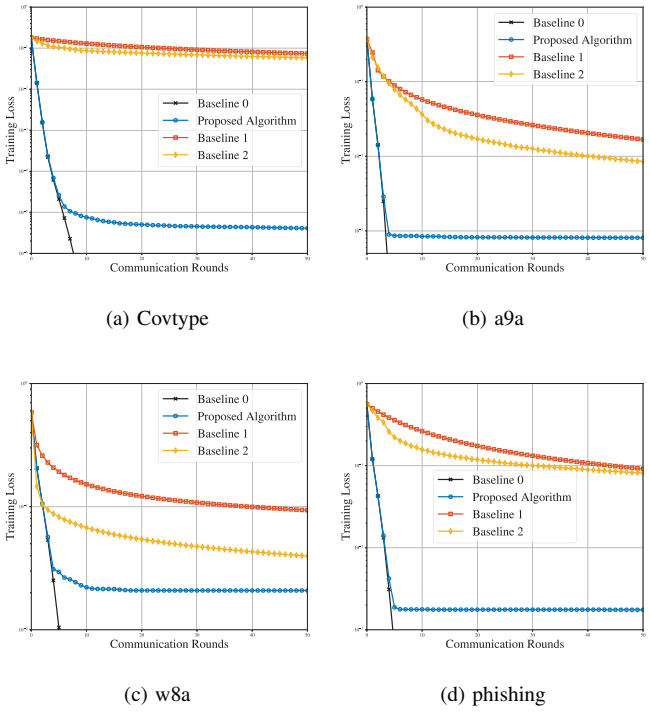
After the identification of the neighboring solution set, the candidate states are also determined according to the sets in $\mathcal{F}^{(k)}$, and we need to choose a state to approach the optimal set. Based on the distribution introduced in [61], we sample a device set in $\mathcal{F}^{(k)}$ with the probability

$$\mathbb{P}\left(\tilde{\mathcal{S}}^{(k)}\right) = \frac{\exp\left(-J\left(\tilde{\mathcal{S}}^{(k)}, \tilde{\boldsymbol{a}}^{(k)}\right)/T^{(k)}\right)}{\sum_{\tilde{\mathcal{S}}\in\mathcal{F}^{(k)}}\exp\left(-J\left(\tilde{\mathcal{S}}, \tilde{\boldsymbol{a}}\right)/T^{(k)}\right)}\ , \tag{30}$$

where $J(\boldsymbol{x}, \boldsymbol{y})$ denotes the objective function value of $\mathscr{P}$. Here, the receiver beamforming vector is calculated through DCA with the given set of selected devices.

In the distribution (30), there is a special parameter $T^{(k)}$ serving as the temperature. The algorithm starts from a relatively high temperature $T^{(0)}$ in order to move around the solution space freely, rather than being stuck in a local minimum point. As the algorithm proceeds, the algorithm slowly decreases the temperature by the factor $\rho$ to focus on the states that minimize the objective function. Besides, to reduce the computational complexity, we have adopted a similar warm start technique as in [33]. The optimal beamforming vector in the previous iteration is used to serve as the initial point to accelerate the process of beamforming

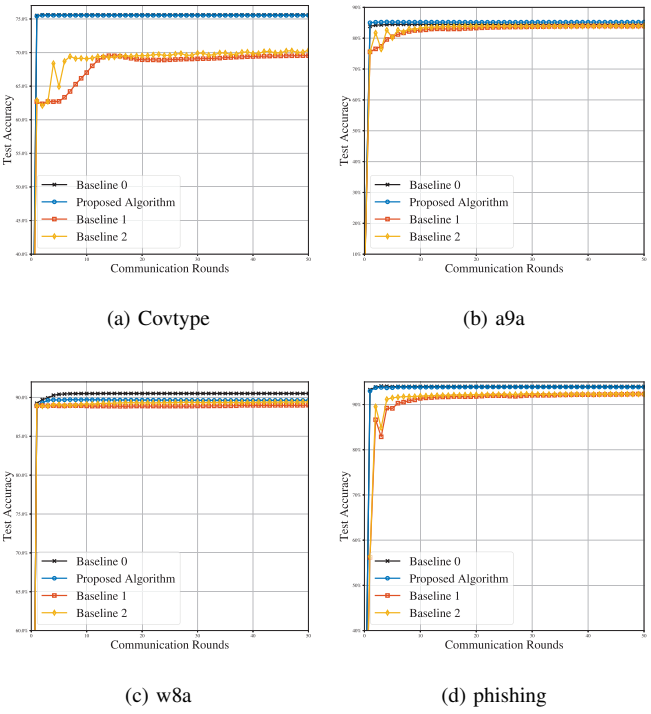
optimization. The overall process of system optimization is outlined in Algorithm 3.




(a) Covtype (b) a9a

(c) w8a (d) phishing

Fig. 2: Training loss of the proposed algorithm and two first-order algorithms.




(a) Covtype (b) a9a

(c) w8a (d) phishing

Fig. 3: Test accuracy of the proposed algorithm and two first-order algorithms.

## V. Simulation Results

In this section, we evaluate the performance of the proposed schemes to demonstrate the advantage of our proposed second-order federated optimization algorithm and the effectiveness of our system optimization approach. Code for our experiments are available at: https://github.com/Golden-Slumber/AirFL-2nd. We first consider logistic regression with the loss function of the $i$-th device $F_i(\boldsymbol{w}) = \frac{1}{|\mathcal{D}_i|}\sum_{\boldsymbol{z}_{i,j}=(\boldsymbol{u}_{i,j},v_{i,j})\in\mathcal{D}_i}\log\left(1+\exp\left(-v_{i,j}\boldsymbol{u}_{i,j}^{\mathsf{T}}\boldsymbol{w}\right)\right) + \frac{\gamma}{2}\|\boldsymbol{w}\|_2^2$, where the regularization parameter is set to be $\gamma = 10^{-8}$. As for datasets, we adopt four different standard datasets from the LIBSVM library: Covtype, a9a, w8a, and phishing. In this paper, we consider a distributed wireless scenario, where these data samples are uniformly distributed in $m = 20$ devices, the server is equipped with $k = 5$ antennas. The channel coefficients are given by the small-scale fading coefficients $\{\boldsymbol{h}'_{t,i}\}$ multiplied by the path loss gain $PL_i$, i.e., $\boldsymbol{h}_{t,i} = PL_i\boldsymbol{h}'_{t,i}$. Here, the small-scale fading coefficients follow the i.i.d complex normal distribution $\mathcal{CN}(0,\boldsymbol{I})$. The path loss gain is given by $PL_i = \sqrt{G_0}\left(d_0/d_i\right)^{\nu/2}$, where $G_0 = 10^{-3.35}$ is the average channel power gain with the distance to the server $d_0 = 1$ m, $d_i \in [100, 120]$ stands for the distance between the $i$-th device and the server, and $\nu = 3.76$ represents the path loss exponent factor. For the step size $\alpha$, we use backtracking line search to find $\alpha$ satisfying the Armijo–Goldstein condition [46, Chapter 3]. For the system optimization, we set $\lambda = 0.1$, $\tilde{\delta} = 0.01$, penalty factor $\theta = 1$, threshold $\xi = 10^{-10}$, initial temperature $T_0 = 100$, $\rho = 0.9$, and $K = 30$. Besides, we use Baseline 0 to denote the centralized training setting in all experiments.

Furthermore, we also consider an image classification problem on a non-i.i.d dataset constructed from the Fashion-MNIST dataset at the end of this section. To address it, we train a softmax classifier with cross-entropy loss and $\ell_2$ regularization term. To be specific, the loss function of the $i$-th device is given as $F_i(\boldsymbol{W}) = \frac{1}{|\mathcal{D}_i|}\sum_{(\boldsymbol{u},v)\in\mathcal{D}_i}\sum_{c=1}^{C}\mathbf{1}\{v = c\}\log\frac{\exp(\boldsymbol{u}^{\mathsf{T}}\boldsymbol{w}_c)}{\sum_{j=1}^{C}\exp(\boldsymbol{u}^{\mathsf{T}}\boldsymbol{w}_j)} + \frac{\gamma}{2}\sum_{c=1}^{C}\|\boldsymbol{w}_c\|_2^2$, where $\boldsymbol{W} = [\boldsymbol{w}_1,\ldots,\boldsymbol{w}_C]$ is the concatenation of parameter vectors related to different classes, and $C = 10$ represents the total number of classes.

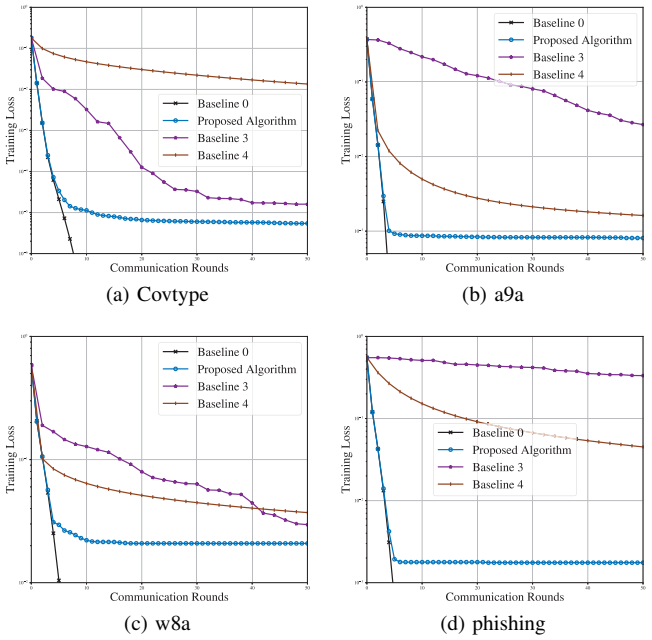
### A. Comparison with First-Order Algorithms

We compared our proposed algorithm with two existing AirComp-based first-order algorithms in this experiment, where SNR is set to 80 dB:

1) Baseline 1: AirComp-based Federated Averaging (FedAvg) algorithm with DC-based optimization framework [29], where the threshold of MSE is set to 5 dB.
2) Baseline 2: AirComp-based Fedsplit algorithm [62], where the threshold for device selection is set to 0.5.

Fig. 2 and Fig. 3 show the performance of these algorithms in training loss and test accuracy. Regarding the optimality gap, benefiting from the linear-quadratic convergence rate, the proposed algorithm reaches a small optimality gap in the first few dozen communication rounds, while that of the first-order methods remains at a relatively higher level. As for

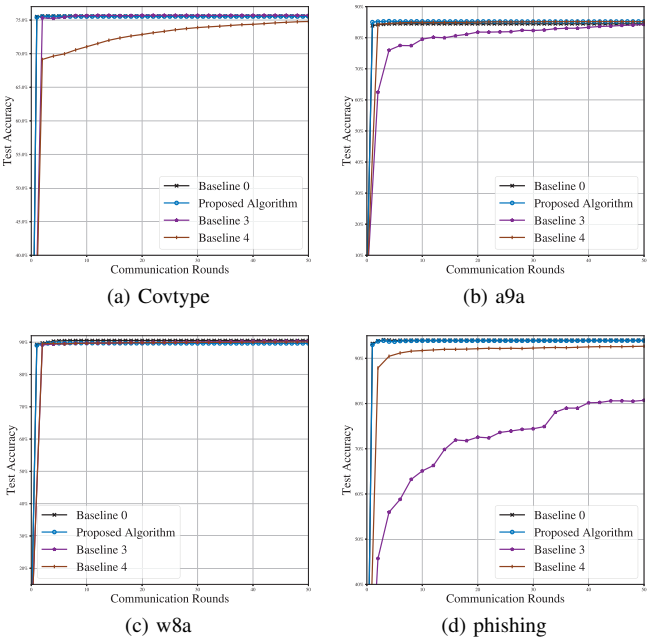
the test accuracy, our proposed algorithm can quickly reach and stabilize at a high accuracy level, while the first-order methods have relatively low and fluctuating accuracy. Overall, our proposed algorithm keeps a quadratic convergence rate at the beginning of FL process, resulting in fewer communication rounds to complete the learning task than first-order algorithms. This further leads to less wireless channel impact and better learning performance, as illustrated in the simulation results.




(a) Covtype (b) a9a

(c) w8a (d) phishing

Fig. 4: Training loss of the proposed algorithm and two second-order algorithms.




(a) Covtype (b) a9a

(c) w8a (d) phishing

Fig. 5: Test accuracy of the proposed algorithm and two second-order algorithms.

### B. Comparison with Second-Order Algorithms

In this experiment, we compared our proposed algorithm with the following two state-of-the-art second-order algorithms under over-the-air computation:

1) Baseline 3: GIANT [18] with over-the-air computation. GIANT requires an extra aggregation of local gradients, leading to two communication rounds in each iteration. The communication model of this gradients aggregation is implemented in the same way of $\boldsymbol{p}_t$, as illustrated in Section II-C. Here, we set $|\mathcal{S}_t| = m$, and the receiver beamforming vector is optimized through DCA.
2) Baseline 4: DANE [16] with over-the-air computation. Similar to GIANT, It also requires an aggregation of local gradients, so its implementation is the same as GIANT.

Fig. 4 and Fig. 5 plot the training loss and the test accuracy, respectively, where SNR is set to 70 dB. It is observed that our proposed algorithm converges faster and remains stable at a relatively high level of accuracy, while the compared methods, AirComp-based GIANT and AirComp-based DANE, have a slower convergence rate. This is because both the procedures of GIANT and DANE involve aggregating local gradients to calculate the global gradient in each iteration. Comparatively, our proposed algorithm avoids such transmission by the use of local Newton step aggregation, which leads to a relatively better convergence rate. Thus, we can see that our proposed algorithm outperforms AirComp-based GIANT and AirComp-based DANE.

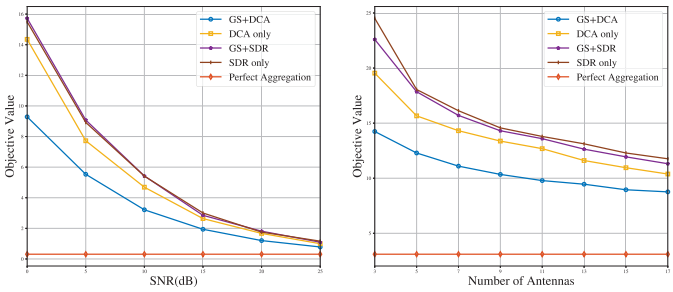



Fig. 6: Objective value of system optimization problem $\mathscr{P}$ versus SNR and number of antennas.

### C. Effectiveness of Proposed System Optimization Approach

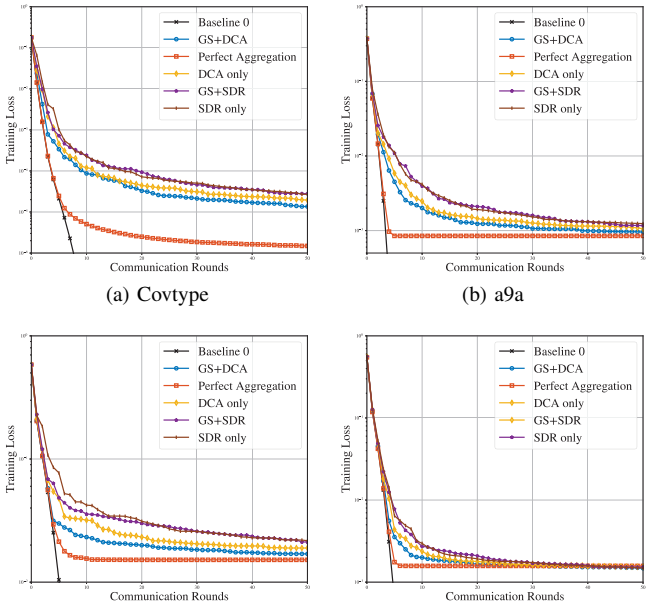
In this experiment, we evaluated the performance using GS+DCA to accomplish system optimization with four settings:

1) perfect aggregation, where the model is aggregated without wireless channel impact.
2) GS+SDR, where the receiver beamforming optimization is performed through SDR.
3) DCA only, where we only perform beamforming optimization through DCA.
4) SDR only, where we only perform beamforming optimization through SDR.

To verify the effectiveness of the device selection, we consider the distance heterogeneity and data size heterogeneity in this

experiment. Specifically, as for distance heterogeneity, we set the distance of $10\%$ devices to be $d_i \in [200, 220]$ while the rest to be $d_i \in [50, 60]$. As for data size heterogeneity, we set the data size of $10\%$ devices to be $|\mathcal{D}_i| \in [0.008\frac{n}{m}, 0.01\frac{n}{m}]$ while the rest to be $|\mathcal{D}_i| \in [1.01\frac{n}{m}, 1.11\frac{n}{m}]$.

We first numerically evaluate the objective value of the system optimization problem $\mathscr{P}$ under different settings in Fig. 6 by averaging 100 channel realizations. The objective value of perfect aggregation does not depend on SNR and the number of antennas since the error during the FL process in this situation only comes from the approximation as (27) indicates. The objective values of all settings decrease as SNR and the number of antennas increase, due to the mitigation of noise effect and the increase of diversity gain [53], respectively. However, the objective value of GS+DCA is smaller than that of other settings. On the one hand, SDR fails to give a precise solution for the receiver beamforming vector as the size of the problem grows. This further leads to the ineffectiveness of device selection in GS+SDR and worse performance compared with the settings using DCA to perform beamforming optimization. On the other hand, device selection in GS+DCA mitigates the straggler issue caused by distance heterogeneity and data size heterogeneity, resulting in a better performance compared with DCA only.

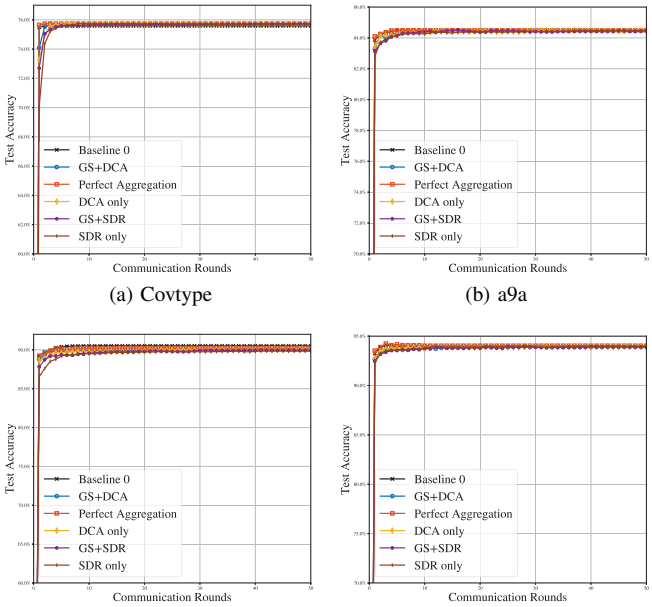



(a) Covtype (b) a9a

(c) w8a (d) phishing

Fig. 7: Training loss of the proposed algorithm in different system optimization settings.

Fig. 7 plots the training loss for our proposed algorithm in different system optimization settings, where SNR is set to 35 dB. The results show that with device selection and a more precise solution given by DCA, the error term can be minimized in each iteration and a smaller optimality gap close to that of perfect aggregation can be obtained. As revealed in Fig. 8, this smaller optimality gap further leads to higher test accuracy, demonstrating that our proposed system optimization approach effectively improves learning performance.




(a) Covtype (b) a9a

(c) w8a (d) phishing

Fig. 8: Test accuracy of the proposed algorithm in different system optimization settings.

### *D. Fashion-MNIST Data Set*

We consider an image classification problem on a non-i.i.d dataset constructed from the Fashion-MNIST dataset in this experiment, where $m = 10$ and SNR is set to 90 dB. The related parameters are set to be the same as the previous experiments, and we use the percentage of correctly classified test images to evaluate the learning performance.

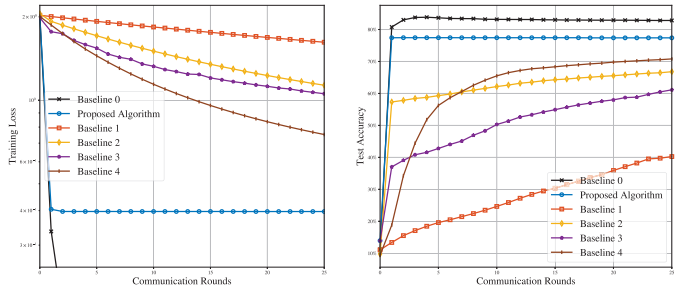



Fig. 9: Simulation results on the Fashion-MNIST dataset.

Fig. 9 presents the training loss and test accuracy versus communication rounds of our proposed algorithm and four baseline algorithms. It reveals that our proposed algorithm significantly outperforms the baseline algorithms. On the one hand, it keeps a better convergence rate than first-order algorithms, leading to fewer communication rounds between the devices and the server. On the other hand, compared with other second-order algorithms under over-the-air computation, the aggregation operation only occurs once per iteration in our proposed algorithm. Therefore, our proposed algorithm is more communication-efficient than baseline algorithms in terms of both the total iteration rounds and the communication

within each iteration, which further benefit learning performance, as illustrated in Fig. 9.

## VI. Conclusion

In this paper, we developed a communication-efficient FL system by over-the-air second-order federated optimization algorithm. The communication rounds and communication latency at each round can be simultaneously reduced. This is achieved by leveraging the second-order information of the learning loss function for achieving fast convergence rates and exploiting the signal superposition property of a multiple access channel for fast model aggregation. The characterized convergence behavior reveals a linear-quadratic convergence rate for the proposed algorithm. As the proposed algorithm is accompanied by an accumulative error term in each iteration, a system optimization problem was formulated to minimize the total error gap while achieving a precise model. We then presented Gibbs Sampling and DC programming methods to jointly optimize device selection and receiver beamforming. The experimental results illustrated that our proposed algorithm and network optimization approach can achieve high communication efficiency for FL systems.

## Appendix A
## Proof of Lemma 3

Inspired by the analysis in [41], to bound $\hat{\boldsymbol{p}}_t$ through $\boldsymbol{p}^*$, the difference between the values of their quadratic functions is essential. According to (21), here we decompose this difference as

$$\begin{aligned}
&\phi_t\left(\hat{\boldsymbol{p}}_t\right)-\phi_t\left(\boldsymbol{p}^*\right)\\
=&\frac{1}{2}\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left(\hat{\boldsymbol{p}}_t-\boldsymbol{p}^*\right)\right\|^2\\
=&\frac{1}{2}\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left[\left(\bar{\boldsymbol{p}}_t-\boldsymbol{p}^*\right)+\left(\boldsymbol{p}_t-\bar{\boldsymbol{p}}_t\right)+\left(\tilde{\boldsymbol{p}}_t-\boldsymbol{p}_t\right)+\left(\hat{\boldsymbol{p}}_t-\tilde{\boldsymbol{p}}_t\right)\right]\right\|^2\\
\leq&\underbrace{\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left(\boldsymbol{p}_t-\bar{\boldsymbol{p}}_t\right)\right\|^2}_{\text{Term 1}}+\underbrace{3\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left(\bar{\boldsymbol{p}}_t-\boldsymbol{p}^*\right)\right\|^2}_{\text{Term 2}}\\
&+\underbrace{3\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left(\tilde{\boldsymbol{p}}_t-\boldsymbol{p}_t\right)\right\|^2}_{\text{Term 3}}+\underbrace{3\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left(\hat{\boldsymbol{p}}_t-\tilde{\boldsymbol{p}}_t\right)\right\|^2}_{\text{Term 4}},
\end{aligned}$$

As for Term 1, by Lemma 1, we have $(1-\lambda)\boldsymbol{M}_t^{\mathsf{T}}\boldsymbol{M}_t \preceq \boldsymbol{M}_t^{\mathsf{T}}\boldsymbol{L}_i\boldsymbol{L}_i^{\mathsf{T}}\boldsymbol{M}_t \preceq (1+\lambda)\boldsymbol{M}_t^{\mathsf{T}}\boldsymbol{M}_t$. Through this we can get $(1-\lambda)\boldsymbol{H}_t \preceq \boldsymbol{H}_{i,t} \preceq (1+\lambda)\boldsymbol{H}_t$. Thus, there exists matrix $\xi_i$ satisfying $\boldsymbol{H}_t^{\frac{1}{2}}\boldsymbol{H}_{t,i}^{-1}\boldsymbol{H}_t^{\frac{1}{2}} = \boldsymbol{I}+\xi_i$ and $-\frac{\lambda}{1+\lambda} \preceq \xi_i \preceq \frac{\lambda}{1-\lambda}$, which leads to a useful property: $\left\|\boldsymbol{H}_t^{\frac{1}{2}}\boldsymbol{H}_{t,i}^{-1}\boldsymbol{H}_t^{\frac{1}{2}}\right\| \leq 1+\frac{\lambda}{1-\lambda}=\frac{1}{1-\lambda}$. With this property and Lemma 2, we can get the following inequality:

$$\begin{aligned}
&\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left(\boldsymbol{p}_t-\bar{\boldsymbol{p}}_t\right)\right\|_2\\
\leq&\frac{1}{n}\sum_{i\in\mathcal{S}}\left|\mathcal{D}_i\right|\left\|\boldsymbol{H}_t^{\frac{1}{2}}\boldsymbol{H}_{t,i}^{-1}\boldsymbol{H}_t^{\frac{1}{2}}\right\|_2\left\|\boldsymbol{H}_t^{-\frac{1}{2}}\left(\boldsymbol{g}_{t,i}-\boldsymbol{g}_t\right)\right\|_2\\
\leq&\frac{1}{1-\lambda}\frac{1}{\sigma_{min}\left(\boldsymbol{H}_t\right)}\frac{1}{n}\\
&\cdot\sum_{i\in\mathcal{S}}\left|\mathcal{D}_i\right|\left(1+\sqrt{2\ln\frac{1}{\delta_i}}\right)\sqrt{\frac{1}{\left|\mathcal{D}_i\right|}}\max_j\left\|\boldsymbol{n}_j\right\|_2\\
\leq&\frac{1}{1-\lambda}\frac{1}{\sigma_{min}\left(\boldsymbol{H}_t\right)}\frac{1}{n}\\
&\cdot\left(1+\sqrt{2\ln\frac{1}{\bar{\delta}}}\right)\max_j\left\|\boldsymbol{n}_j\right\|_2\sqrt{\sum_{i\in\mathcal{S}}m\left|\mathcal{D}_i\right|}\\
=&\frac{1}{1-\lambda}\frac{1}{\sigma_{min}\left(\boldsymbol{H}_t\right)}\left(1+\sqrt{2\ln\frac{1}{\bar{\delta}}}\right)\sqrt{\frac{m}{n}}\max_j\left\|\boldsymbol{n}_j\right\|_2 .
\end{aligned}$$

For convenience, we denote: $\mathcal{G} = \frac{1}{1-\lambda}\frac{1}{\sigma_{min}(\boldsymbol{H}_t)}\left(1+\sqrt{2\ln\frac{1}{\bar{\delta}}}\right)\max_j\left\|\boldsymbol{n}_j\right\|_2$, and Term 1 is bounded by Term 1 $\leq \frac{m}{n}\mathcal{G}^2$. As for Term 2, based on the analysis in [18, Lemma 6], we have

$$\begin{aligned}
\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left(\bar{\boldsymbol{p}}_t-\boldsymbol{p}^*\right)\right\| \leq&\left\|\frac{1}{n}\sum_{i\in\mathcal{S}}\left|\mathcal{D}_i\right|\boldsymbol{H}_t^{\frac{1}{2}}\left(\bar{\boldsymbol{p}}_{t,i}-\boldsymbol{p}^*\right)\right\|_2\\
\leq&\frac{1}{n}\sum_{i\in\mathcal{S}}\left|\mathcal{D}_i\right|\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left(\bar{\boldsymbol{p}}_{t,i}-\boldsymbol{p}^*\right)\right\|_2\\
\leq&\zeta_1\left\|\boldsymbol{H}_t^{\frac{1}{2}}\boldsymbol{p}^*\right\|_2 ,
\end{aligned}$$

with $\zeta_1=\tau\left(\lambda+\frac{\lambda^2}{1-\lambda}\right)$ and $\tau=\frac{\sigma_{\max}\left(\boldsymbol{M}^\top\boldsymbol{M}\right)}{\sigma_{\max}\left(\boldsymbol{M}^\top\boldsymbol{M}\right)+n\gamma}$. Then Term 2 is bound by:

$$\text{Term 2}=3\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left(\bar{\boldsymbol{p}}_t-\boldsymbol{p}^*\right)\right\|^2 \leq 3\zeta_1^2\left\|\boldsymbol{H}_t^{\frac{1}{2}}\boldsymbol{p}^*\right\|_2^2 = -3\zeta_1^2\phi\left(\boldsymbol{p}^*\right).$$

As for Term 3, it can be reformulated as follows:

$$\begin{aligned}
&\text{Term 3}\\
=&3\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left(\tilde{\boldsymbol{p}}_t-\boldsymbol{p}_t\right)\right\|^2\\
=&3\left\|\boldsymbol{H}_t^{\frac{1}{2}}\left(\frac{n-\sum_{i\in\mathcal{S}_t}\left|\mathcal{D}_i\right|}{n\sum_{i\in\mathcal{S}_t}\left|\mathcal{D}_i\right|}\sum_{i\in\mathcal{S}_t}\left|\mathcal{D}_i\right|\boldsymbol{p}_{t,i}-\frac{1}{n}\sum_{i\in\mathcal{S}\setminus\mathcal{S}_t}\left|\mathcal{D}_i\right|\boldsymbol{p}_{t,i}\right)\right\|^2 .
\end{aligned}$$

According to the analysis in [63, Section 3.1], it follows:

$$\begin{aligned}
&\text{Term 3}\\
\leq&12\left(1-\frac{\sum_{i\in\mathcal{S}_t}\left|\mathcal{D}_i\right|}{n}\right)^2\left(\left\|\boldsymbol{H}_t^{\frac{1}{2}}\boldsymbol{p}_{t,i}-\boldsymbol{H}_t^{\frac{1}{2}}\bar{\boldsymbol{p}}_{t,i}\right\|_2\right.\\
&\left.+\left\|\boldsymbol{H}_t^{\frac{1}{2}}\bar{\boldsymbol{p}}_{t,i}-\boldsymbol{H}_t^{\frac{1}{2}}\boldsymbol{p}^*\right\|_2+\left\|\boldsymbol{H}_t^{\frac{1}{2}}\boldsymbol{p}^*\right\|_2\right)^2\\
\overset{(a)}{\leq}&12\left(1-\frac{\sum_{i\in\mathcal{S}_t}\left|\mathcal{D}_i\right|}{n}\right)^2\left(\sqrt{\frac{1}{\min_{i\in\mathcal{S}_t}\left|\mathcal{D}_i\right|}}\mathcal{G}\right.\\
&\left.+\left\|\boldsymbol{H}_t^{\frac{1}{2}}\bar{\boldsymbol{p}}_{t,i}-\boldsymbol{H}_t^{\frac{1}{2}}\boldsymbol{p}^*\right\|_2+\left\|\boldsymbol{H}_t^{\frac{1}{2}}\boldsymbol{p}^*\right\|_2\right)^2\\
\overset{(b)}{\leq}&12\left(1-\frac{\sum_{i\in\mathcal{S}_t}\left|\mathcal{D}_i\right|}{n}\right)^2\left(\sqrt{\frac{1}{\min_{i\in\mathcal{S}_t}\left|\mathcal{D}_i\right|}}\mathcal{G}\right.\\
&\left.+\left(\zeta_1+1\right)\left\|\boldsymbol{H}_t^{\frac{1}{2}}\boldsymbol{p}^*\right\|_2\right)^2\\
\leq&24\left(1-\frac{\sum_{i\in\mathcal{S}_t}\left|\mathcal{D}_i\right|}{n}\right)^2\frac{1}{\min_{i\in\mathcal{S}_t}\left|\mathcal{D}_i\right|}\mathcal{G}^2\\
&-24\vartheta^2\left(\zeta_1+1\right)^2\phi\left(\boldsymbol{p}^*\right)
\end{aligned}$$

where $\zeta_1 = \tau\left(\lambda+\frac{\lambda^2}{1-\lambda}\right)$, $\tau = \frac{\sigma_{\max}\left(\boldsymbol{M}^\top\boldsymbol{M}\right)}{\sigma_{\max}\left(\boldsymbol{M}^\top\boldsymbol{M}\right)+n\gamma}$, $\vartheta = \max_t\left(1-\frac{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}{n}\right) < 1$, $(a)$ and $(b)$ are obtained in the way similar to the analysis of Term 1 and Term 2. As for Term 4, we have:

$$\begin{aligned}\text{Term 4} =& 3\left\|\boldsymbol{H}_t^{\frac{1}{2}}\frac{1}{\left(\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|\right)\sqrt{\eta_t}}\boldsymbol{a}_t^{\mathsf{H}}\boldsymbol{E}_t\right\|_2^2\\ \leq& \frac{3}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left\|\frac{1}{\left(\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|\right)\sqrt{\eta_t}}\boldsymbol{a}_t^{\mathsf{H}}\boldsymbol{E}_t\right\|^2 .\end{aligned}$$

We can get the final result by combining the bound of Term 1, 2, 3 and 4 together:

$$\begin{aligned}&\phi\left(\hat{\boldsymbol{p}}_t\right)-\phi\left(\boldsymbol{p}^*\right)\leq\epsilon^2-\zeta^2\phi\left(\boldsymbol{p}^*\right)\\ \Rightarrow&\phi\left(\boldsymbol{p}^*\right)\leq\phi\left(\hat{\boldsymbol{p}}_t\right)\leq\epsilon^2+\left(1-\zeta^2\right)\phi\left(\boldsymbol{p}^*\right) ,\end{aligned}$$

where $\epsilon$ and $\zeta$ are defined as (24) and (25).

## Appendix B
## Proof of Theorem 1

Based on Lemma 3 and Lemma 4, we have:

$$\begin{aligned}&\boldsymbol{\Delta}_{t+1}^\top\boldsymbol{H}_t\boldsymbol{\Delta}_{t+1}\\ \leq& L\left\|\boldsymbol{\Delta}_{t+1}\right\|\left\|\boldsymbol{\Delta}_t\right\|^2+\frac{\zeta^2}{1-\zeta^2}\boldsymbol{\Delta}_t^\top\boldsymbol{H}_t\boldsymbol{\Delta}_t+2\epsilon^2\\ \leq& L\left\|\boldsymbol{\Delta}_{t+1}\right\|\left\|\boldsymbol{\Delta}_t\right\|^2+\left(\frac{\zeta^2}{1-\zeta^2}\sigma_{\max}\left(\mathbf{H}_t\right)\right)\left\|\boldsymbol{\Delta}_t\right\|^2+2\epsilon^2 .\end{aligned}$$

According to the analysis in [41, Appendix A], this leads to:

$$\begin{aligned}&\left\|\boldsymbol{\Delta}_{t+1}\right\|\\ \leq&\max\left\{\sqrt{\frac{\sigma_{\max}\left(\boldsymbol{H}_t\right)}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left(\frac{\zeta^2}{1-\zeta^2}\right)}\left\|\boldsymbol{\Delta}_t\right\|,\frac{L}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left\|\boldsymbol{\Delta}_t\right\|^2\right\}\\ &+\frac{2\epsilon}{\sqrt{\sigma_{\min}\left(\mathbf{H}_t\right)}} .\end{aligned}\tag{31}$$

As for the error term $\epsilon$, we have:

$$\begin{aligned}&\epsilon\\ =&\left\{\frac{3}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left\|\frac{1}{\left(\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|\right)\sqrt{\eta_t}}\boldsymbol{a}_t^{\mathsf{H}}\boldsymbol{E}_t\right\|^2\right.\\ &\left.+\left[24\left(1-\frac{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}{n}\right)^2\frac{1}{\min_{i\in\mathcal{S}_t}|\mathcal{D}_i|}+\frac{m}{n}\right]\mathcal{G}^2\right\}^{\frac{1}{2}}\\ \leq&\sqrt{\frac{3}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}}\frac{d}{\left(\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|\right)\sqrt{\eta_t}}\left\|\boldsymbol{a}_t\right\|\left\|\boldsymbol{e}_t\right\|\\ &+\sqrt{24\left(1-\frac{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}{n}\right)^2\frac{1}{\min_{i\in\mathcal{S}_t}|\mathcal{D}_i|}+\frac{m}{n}}\cdot\mathcal{G} .\end{aligned}$$

To handle the random variable $\boldsymbol{e}_t$ in $\epsilon$, we take expectations over $\boldsymbol{e}_t$ on both sides of (31) :

$$\begin{aligned}&\mathbb{E}\left(\left\|\boldsymbol{\Delta}_{t+1}\right\|\right)\\ \leq&\max\left\{\sqrt{\frac{\sigma_{\max}\left(\boldsymbol{H}_t\right)}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left(\frac{\zeta^2}{1-\zeta^2}\right)}\left\|\boldsymbol{\Delta}_t\right\|,\frac{L}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left\|\boldsymbol{\Delta}_t\right\|^2\right\}\\ &+\frac{2\sqrt{3}}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\frac{d\left\|\boldsymbol{a}_t\right\|\mathbb{E}\left(\left\|\boldsymbol{e}_t\right\|\right)}{\left(\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|\right)\sqrt{\eta_t}}\\ &+\sqrt{24\left(1-\frac{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}{n}\right)^2\frac{1}{\min_{i\in\mathcal{S}_t}|\mathcal{D}_i|}+\frac{m}{n}}\\ &\cdot\frac{1}{1-\lambda}\frac{2}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left(1+\sqrt{2\ln\left(\frac{1}{\tilde{\delta}}\right)}\right)\max_j\left\|\boldsymbol{n}_j\right\|\\ \overset{(c)}{\leq}&\max\left\{\sqrt{\frac{\sigma_{\max}\left(\boldsymbol{H}_t\right)}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left(\frac{\zeta^2}{1-\zeta^2}\right)}\left\|\boldsymbol{\Delta}_t\right\|,\frac{L}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left\|\boldsymbol{\Delta}_t\right\|^2\right\}\\ &+\frac{2\sqrt{3}}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\frac{d\sigma\left\|\boldsymbol{a}_t\right\|}{\left(\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|\right)\sqrt{\eta_t}}\\ &+\sqrt{24\left(1-\frac{\sum_{i\in\mathcal{S}_t}|\mathcal{D}_i|}{n}\right)^2\frac{1}{\min_{i\in\mathcal{S}_t}|\mathcal{D}_i|}+\frac{m}{n}}\\ &\cdot\frac{1}{1-\lambda}\frac{2}{\sigma_{\min}\left(\boldsymbol{H}_t\right)}\left(1+\sqrt{2\ln\left(\frac{1}{\tilde{\delta}}\right)}\right)\max_j\left\|\boldsymbol{n}_j\right\| .\end{aligned}$$

## References


[1] L. Xu, C. Jiang, J. Wang, J. Yuan, and Y. Ren, "Information security in big data: privacy and data mining," *Ieee Access*, vol. 2, pp. 1149–1176, 2014.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, pp. 1273–1282, PMLR, 2017.

[3] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans Intell*, vol. 10, no. 2, pp. 1–19, 2019.

[5] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[6] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data, *et al.*, "A field guide to federated optimization," *arXiv preprint arXiv:2107.06917*, 2021.

[7] Y. Shi, K. Yang, T. Jiang, J. Zhang, and K. B. Letaief, "Communication-efficient edge ai: Algorithms and systems," *IEEE Commun. Surv*, vol. 22, no. 4, pp. 2167–2191, 2020.

[8] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signsgd: Compressed optimisation for non-convex problems," in *ICML*, pp. 560–569, PMLR, 2018.

[9] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv preprint arXiv:1704.05021*, 2017.

[10] X. Chen, S. Liu, K. Xu, X. Li, X. Lin, M. Hong, and D. Cox, "Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization," *Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 7204–7215, 2019.

[11] H. Gao and H. Huang, "Can stochastic zeroth-order frank-wolfe method converge faster for non-convex problems?," in *ICML*, pp. 3377–3386, PMLR, 2020.

[12] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *FoCM*, vol. 17, no. 2, pp. 527–566, 2017.

[13] B. Woodworth, K. K. Patel, S. Stich, Z. Dai, B. Bullins, B. Mcmahan, O. Shamir, and N. Srebro, "Is local sgd better than minibatch sgd?," in *ICML*, pp. 10334–10343, PMLR, 2020.

[14] H. Yuan and T. Ma, "Federated accelerated stochastic gradient descent," *arXiv preprint arXiv:2006.08950*, 2020.

[15] S. Bischoff, S. Günnemann, M. Jaggi, and S. U. Stich, "On second-order optimization methods for federated learning," *arXiv preprint arXiv:2109.02388*, 2021.

[16] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *ICML*, pp. 1000–1008, PMLR, 2014.

[17] V. Smith, S. Forte, M. Chenxin, M. Takáč, M. I. Jordan, and M. Jaggi, "Cocoa: A general framework for communication-efficient distributed optimization," *J Mach Learn Res*, vol. 18, p. 230, 2018.

[18] S. Wang, F. Roosta, P. Xu, and M. W. Mahoney, "Giant: Globally improved approximate newton method for distributed optimization," in *Adv. Neural Inf. Process. Syst.*, pp. 2332–2342, 2018.

[19] R. Crane and F. Roosta, "Dingo: Distributed newton-type method for gradient-norm optimization," *arXiv preprint arXiv:1901.05134*, 2019.

[20] R. Crane and F. Roosta, "Dino: Distributed newton-type optimization method," in *ICML*, pp. 2174–2184, PMLR, 2020.

[21] Y. Zhang and X. Lin, "Disco: Distributed optimization for self-concordant empirical loss," in *ICML*, pp. 362–370, PMLR, 2015.

[22] K. Yang, Y. Shi, Y. Zhou, Z. Yang, L. Fu, and W. Chen, "Federated machine learning for intelligent iot via reconfigurable intelligent surface," *IEEE Network*, vol. 34, no. 5, pp. 16–22, 2020.

[23] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6g: Ai empowered wireless networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, 2019.

[24] L. Li, L. Yang, X. Guo, Y. Shi, H. Wang, W. Chen, and K. B. Letaief, "Delay analysis of wireless federated learning based on saddle point approximation and large deviation theory," *arXiv preprint arXiv:2103.16994*, 2021.

[25] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, 2020.

[26] A. Elgabli, J. Park, C. B. Issaid, and M. Bennis, "Harnessing wireless channels for scalable and privacy-preserving federated learning," *IEEE Trans Commun*, 2021.

[27] W.-T. Chang and R. Tandon, "Communication efficient federated learning over multiple access channels," *arXiv preprint arXiv:2001.08737*, 2020.

[28] B. Nazer and M. Gastpar, "Computation over multiple-access channels," *IEEE Trans. Inf. Theory*, vol. 53, no. 10, pp. 3498–3516, 2007.

[29] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 3, pp. 2022–2035, 2020.

[30] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 1, pp. 491–506, 2019.

[31] T. Sery and K. Cohen, "On analog gradient descent learning over multiple access fading channels," *IEEE Trans. Signal Process*, vol. 68, pp. 2897–2911, 2020.

[32] D. Liu and O. Simeone, "Privacy for free: Wireless federated learning via uncoded transmission with adaptive power control," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 170–185, 2020.

[33] H. Liu, X. Yuan, and Y.-J. A. Zhang, "Reconfigurable intelligent surface enabled federated learning: A unified communication-learning design approach," *arXiv preprint arXiv:2011.10282*, 2020.

[34] C. Xiaowen, Z. Guangxu, X. Jie, W. Zhiqin, and C. Shuguang, "Optimized power control design for over-the-air federated edge learning," *arXiv preprint arXiv:2106.09316*, 2021.

[35] C. Xu, S. Liu, Z. Yang, Y. Huang, and K.-K. Wong, "Learning rate optimization for federated learning exploiting over-the-air computation," *arXiv preprint arXiv:2102.02946*, 2021.

[36] G. Zhu, Y. Du, D. Gündüz, and K. Huang, "One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis," *IEEE Trans. Wirel. Commun*, vol. 20, no. 3, pp. 2120–2135, 2020.

[37] X. Wei and C. Shen, "Federated learning over noisy channels: Convergence analysis and design examples," *IEEE Transactions on Cognitive Communications and Networking*, 2022.

[38] X. Fan, Y. Wang, Y. Huo, and Z. Tian, "Joint optimization of communications and federated learning over the air," *arXiv preprint arXiv:2104.03490*, 2021.

[39] H. Guo, Y. Zhu, H. Ma, V. K. Lau, K. Huang, X. Li, H. Nong, and M. Zhou, "Over-the-air aggregation for federated learning: Waveform superposition and prototype validation," *Journal of Communications and Information Networks*, vol. 6, no. 4, pp. 429–442, 2021.

[40] Z. Wang, J. Qiu, Y. Zhou, Y. Shi, L. Fu, W. Chen, and K. B. Lataief, "Federated learning via intelligent reflecting surface," *arXiv preprint arXiv:2011.05051*, 2020.

[41] A. Ghosh, R. K. Maity, and A. Mazumdar, "Distributed newton can communicate less and resist byzantine workers," *arXiv preprint arXiv:2006.08737*, 2020.

[42] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE PAMI*, no. 6, pp. 721–741, 1984.

[43] P. D. Tao and L. T. H. An, "Convex analysis approach to dc programming: theory, algorithms and applications," *Acta mathematica vietnamica*, vol. 22, no. 1, pp. 289–355, 1997.

[44] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*, pp. 421–436, Springer, 2012.

[45] T. Vogels, S. P. Karimireddy, and M. Jaggi, "PowerSGD: Practical low-rank gradient compression for distributed optimization," *Adv. Neural Inf. Process. Syst. 32 (Nips 2019)*, vol. 32, no. CONF, 2019.

[46] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[47] M. M. Amiri and D. Gündüz, "Federated learning over wireless fading channels," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 5, pp. 3546–3557, 2020.

[48] G. Zhu and K. Huang, "Mimo over-the-air computation for high-mobility multimodal sensing," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6089–6103, 2018.

[49] H. Guo, A. Liu, and V. K. Lau, "Analog gradient aggregation for federated learning over wireless networks: Customized design and convergence analysis," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 197–210, 2020.

[50] M. Seif, R. Tandon, and M. Li, "Wireless federated learning with local differential privacy," in *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 2604–2609, IEEE, 2020.

[51] Z. Wang, Y. Shi, Y. Zhou, H. Zhou, and N. Zhang, "Wireless-powered over-the-air computation in intelligent reflecting surface-aided iot networks," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1585–1598, 2020.

[52] W. Fang, Y. Jiang, Y. Shi, Y. Zhou, W. Chen, and K. B. Letaief, "Over-the-air computation via reconfigurable intelligent surface," *arXiv preprint arXiv:2105.05113*, 2021.

[53] L. Chen, X. Qin, and G. Wei, "A uniform-forcing transceiver design for over-the-air function computation," *IEEE Wireless Commun. Lett.*, vol. 7, no. 6, pp. 942–945, 2018.

[54] P. Drineas and M. W. Mahoney, "Randnla: randomized numerical linear algebra," *Communications of the ACM*, vol. 59, no. 6, pp. 80–90, 2016.

[55] D. P. Woodruff, "Sketching as a tool for numerical linear algebra," *arXiv preprint arXiv:1411.4357*, 2014.

[56] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 1, pp. 269–283, 2020.

[57] Z.-Q. Luo, N. D. Sidiropoulos, P. Tseng, and S. Zhang, "Approximation bounds for quadratic optimization with homogeneous quadratic constraints," *SIAM J. Optim*, vol. 18, no. 1, pp. 1–28, 2007.

[58] N. D. Sidiropoulos, T. N. Davidson, and Z.-Q. Luo, "Transmit beamforming for physical-layer multicasting," *IEEE Trans. Signal Process.*, vol. 54, no. 6, pp. 2239–2251, 2006.

[59] S. Hua, K. Yang, and Y. Shi, "On-device federated learning via second-order optimization with over-the-air computation," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pp. 1–5, IEEE, 2019.

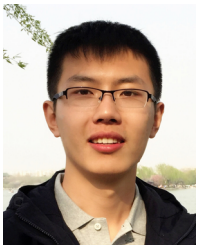
[60] K. Khamaru and M. Wainwright, "Convergence guarantees for a class of non-convex and non-smooth optimization problems," in *International Conference on Machine Learning*, pp. 2601–2610, PMLR, 2018.

[61] P. Brémaud, *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, vol. 31. Springer Science & Business Media, 2013.

[62] S. Xia, J. Zhu, Y. Yang, Y. Zhou, Y. Shi, and W. Chen, "Fast convergence algorithm for analog federated learning," *arXiv preprint arXiv:2011.06658*, 2020.

[63] M. P. Friedlander and M. Schmidt, "Hybrid deterministic-stochastic methods for data fitting," *SIAM J Sci Comput*, vol. 34, no. 3, pp. A1380–A1405, 2012.

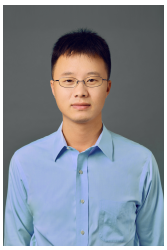

**Peng Yang** received his Bachelor degree in Software Engineering from East China Normal University (ECNU), Shanghai, China, in 2020. He is currently pursuing his Master degree at Software Engineering Institute, ECNU, Shanghai, China. His research interests include federated learning, edge intelligence, and machine learning systems.



**Yuning Jiang** (Member, IEEE) received the B.Sc. degree in electrical engineering from Shandong University, Jinan, China, in 2014, and the Ph.D. degree in information engineering from ShanghaiTech University, Shanghai, China, and the University of Chinese Academy of Sciences, Beijing, China, in 2020. He has ever been a Visiting Scholar with the University of California at Berkeley (UC Berkeley), Berkeley, CA, USA, the University of Freiburg, Freiburg im Breisgau, Germany, and Technische Universität Ilmenau (TU Ilmenau), Ilmenau, Germany, during his Ph.D. study. He is currently a Post-Doctoral Researcher with the Laboratoire d'Automatique, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. His research interests include distributed and federated optimization, robust and stochastic optimization, and model predictive control, particularly for power and energy systems, cyber-physical system, intelligent transportation and wireless communication.

**Ting Wang** received the Ph.D. degree in Computer Science and Engineering from Hong Kong University of Science and Technology, Hong Kong, China, in 2015. He is currently an associate professor with the Software Engineering Institute, East China Normal University, Shanghai, China. Prior to joining ECNU in 2020, he worked at the Bell Labs as a research scientist from 2015 to 2016, and at Huawei as a senior engineer from 2016 to 2020. He is currently an associate editor of IEEE Access, the founding editor-in-chief of IITCIB, and a technical committee member of Computer Communications, Elsevier. His research interests include cloud/edge computing, data center networks, machine learning, and AI-aided intelligent networking.

**Yong Zhou** (Member, IEEE) received the B.Sc. and M.Eng. degrees from Shandong University, Jinan, China, in 2008 and 2011, respectively, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2015. From Nov. 2015 to Jan. 2018, he worked as a postdoctoral research fellow in the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada. He is currently an Assistant Professor in the School of Information Science and Technology, ShanghaiTech University, Shanghai, China. He was the track co-chair of IEEE VTC 2020 Fall and is the general co-chair of IEEE ICC 2022 workshop on edge artificial intelligence for 6G. His research interests include 6G communications, edge intelligence, and Internet of Things.



**Yuanming Shi** (S'13-M'15-SM'20) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2011. He received the Ph.D. degree in electronic and computer engineering from The Hong Kong University of Science and Technology (HKUST), in 2015. Since September 2015, he has been with the School of Information Science and Technology in ShanghaiTech University, where he is currently a tenured Associate Professor. He visited University of California, Berkeley, CA, USA, from October 2016 to February 2017. His research areas include optimization, machine learning, wireless communications, and their applications to 6G, IoT, and edge AI. He was a recipient of the 2016 IEEE Marconi Prize Paper Award in Wireless Communications, the 2016 Young Author Best Paper Award by the IEEE Signal Processing Society, and the 2021 IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award. He is also an editor of IEEE Transactions on Wireless Communications, IEEE Journal on Selected Areas in Communications, and Journal of Communications and Information Networks.

**Colin N. Jones** has been an Associate Professor in the Automatic Control Laboratory at the EPFL in Switzerland since 2017 and an assistant professor from 2011. He was a Senior Researcher at the Automatic Control Lab at ETH Zurich until 2010 and obtained a Ph.D. in 2005 from the University of Cambridge for his work on polyhedral computational methods for constrained control. Prior to that, he was at the University of British Columbia in Canada, where he took his bachelor and master degrees in Electrical Engineering and Mathematics. He is the author or coauthor of more than 200 publications and was awarded an ERC starting grant to study the optimal control of building networks. His current research interests are in the areas of high-speed predictive control and optimization, as well as the control of green energy generation, distribution and management.