DONE: Distributed Approximate Newton-type Method for Federated Edge Learning

Canh T. Dinh, Nguyen H. Tran, *Senior Member, IEEE*, Tuan Dung Nguyen, Wei Bao, *Member, IEEE*, Amir Rezaei Balef, Bing B. Zhou, *Member, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—There is growing interest in applying distributed machine learning to edge computing, forming *federated edge learning*. Federated edge learning faces non-i.i.d. and heterogeneous data, and the communication between edge workers, possibly through distant locations and with unstable wireless networks, is more costly than their local computational overhead. In this work, we propose DONE, a distributed approximate Newton-type algorithm with fast convergence rate for communication-efficient federated edge learning. First, with strongly convex and smooth loss functions, DONE approximates the Newton direction in a distributed manner using the classical Richardson iteration on each edge worker. Second, we prove that DONE has linear-quadratic convergence and analyze its communication complexities. Finally, the experimental results with non-i.i.d. and heterogeneous data show that DONE attains a comparable performance to Newton's method. Notably, DONE requires fewer communication iterations compared to distributed gradient descent and outperforms DANE, FEDL, and GIANT, state-of-the-art approaches, in the case of non-quadratic loss functions.

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

Traditional centralized machine learning procedures are gradually becoming inadequate as the computational and storage capacities of individual machines fall short of the data quantity involved in learning. The shift to intelligence at the edge [1], [2], [3], including parallel and distributed approaches, are more capable, scalable and can be set up in various geographical locations.

The existence of (i) powerful edge computing with data abundance and (ii) successful data center-type distributed machine learning architectures raises a natural question: *Can we apply large-scale distributed machine learning to edge computing networks*? We identify two key obstacles to overcome. First, unlike the data sources of data center-type distributed machine learning, which are centrally collected, shuffled and hence homogenous and i.i.d., data at edge networks are collected separately, and therefore heterogeneous and non-i.i.d., similar to cross-device federated learning [4], [5], [6]; thus distributed learning at edge networks is called *federated edge learning*. Second, we see that data center-type distributed machine learning and cross-device federated learning are two extremes: while the former involves learning with optimized computing nodes, data shuffling, and communication networks, the latter requires a massive number of participating devices with constrained computing, storage, and communication capacities. On the other hand, the storage and computing capacities of federated edge learning are comparable to the former, whereas its communication environments are similar to those of the latter due to notable physical distances between edge workers, multi-hop transmission, and different types of communication medium (e.g., wireless/optical/wireline/mmWave.) Compared with local computation cost at each edge worker, the cost of edge communication is considerably higher in terms of speed, delay, and energy consumption [1], [3], [7], [8], [9]; hence, it is often considered as a bottleneck specific to federated edge learning.

Our goal is to develop a distributed algorithm which uses a minimal number of communication rounds to reach a certain precision for convergence and handles non-i.i.d. and heterogeneous data across edge workers.[1] In the literature, first-order methods, which only use gradient information, can be robustly implemented in a distributed manner and require little local computation, are found most frequently [10]. Second-order counterparts like the Newton's method, which have also been well studied in literature [11], [12], use both gradient and curvature information.

- *Canh T. Dinh, Nguyen H. Tran, Wei Bao, Bing B. Zhou, and Albert Y. Zomaya are with the School of Computer Science, The University of Sydney, Sydney, NSW 2006, Australia. E-mail: {canh.dinh, nguyen.tran, wei.bao, bing.zhou, albert.zomaya}@sydney.edu.au.*
- *Tuan Dung Nguyen is with the School of Computing, The Australian National University, Canberra, ACT 2601, Australia. E-mail: josh.nguyen@anu.edu.au.*
- *Amir Rezaei Balef is with the Sharif University of Technology, Tehran 1458889694, Iran. E-mail: amirbalef@gmail.com.*

Manuscript received 23 July 2021; revised 13 Jan. 2022; accepted 18 Jan. 2022. Date of publication 27 Jan. 2022; date of current version 23 May 2022.
This work was supported in part by the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant 102.02-2019.321.
(Corresponding authors: Canh T. Dinh and Nguyen H. Tran.)
Recommended for acceptance by A. J. Peña, M. Si, and J. Zhai.
Digital Object Identifier no. 10.1109/TPDS.2022.3146253

1. An important assumption in many machine learning methods is about independent and identically distributed (i.i.d.) data sample: all data observations are assumed to follow from the same probability distribution and are mutually independent. However, due to the nature of distributed learning in FL, this assumption may be violated (non-i.i.d. data).

Their advantage is that by finding a "better" descent direction, they can lead to a substantially faster convergence rate, therefore requiring much fewer communication rounds to convergence.

In federated edge learning, the cost of local computation required by the (approximate) Newton-type method is gradually remedied by edge workers' improvements in computing power. Although the Newton-type method is not friendly to distributed implementations due to its descent direction structure involving complicated inverse matrix-vector product, we design an algorithm that can overcome this issue and is summarized as follows.

- We propose DONE, a Distributed apprOximate Newton-type mEthod for federated edge learning. When the loss function of the learning task is strongly convex and smooth, DONE *exploits the classical Richardson iteration* to enable the Newton's method to be distributively implementable on the edge workers. With large enough local rounds, DONE generates an effective approximation to the Newton direction. In the literature, we are the first to approximate the Newton direction in a distributed setting using the Richardson iteration.
- Theoretically, we show that DONE has a *global linear-quadratic convergence rate*. We also analyze the computation and communication complexities of DONE. Specifically, with generalized linear models, the computation complexity of DONE is comparable to that of the first-order methods. DONE also never requires the explicit computation of the Hessian, but only needs the Hessian-vector product throughout, reducing the space requirement during optimization.
- Experimentally, in non-i.i.d. and heterogeneous settings, we show that DONE achieves *close performance of the true Newton's method in the same hyperparameter settings* and, therefore, significantly reduces the number of communication iterations compared to the standard first-order gradient descent (GD). DONE also gains performance from DANE [13], FEDL [14], and GIANT [15], the current popular distributed approximate Newton's algorithms for federated edge learning.

1 Related Work

First-Order Distributed Optimization Methods. First-order methods, which find a descent direction based on gradient information, are, paralellizable, easy to implement, and are the most common in practice. Stochastic gradient descent (SGD) [16], accelerated SGD [17], variance reduction SGD [18], [19], stochastic coordinate descent [20], and dual coordinate ascent [21] are examples of this type of algorithms. The trade-off for less computation at the edge, i.e., finding only the gradient and possibly some additional information, is the requirement of many iterations until convergence. Communication overheads become a bigger problem in multi-hop edge networks where the bandwidth varies or is limited among workers. [22], [23] apply scheduling policies while [24] uses compression techniques to alleviate the communication pressure, but they base on the first-order gradient method.

Newton-Type Distributed Optimization Methods. Different from first-order methods, Newton-type methods use not only gradient but also curvature information to find a descent direction. Existing distributed Newton-type algorithms include DANE [13], AIDE [19], DiSCO [25], and GIANT [15]. DANE requires each worker to solve a well-designed local optimization problem, whereas GIANT [15] uses the harmonic mean (instead of the true arithmetic mean) of the Hessian matrices. On the other hand, DiSCO is an inexact damped Newton method using a preconditioned conjugate gradient algorithm, which requires multiple communication exchanges between the workers to find the Newton's direction. Our proposed DONE is another type of distributed *approximate* Newton method that is based on Richardson's iteration for local update.

Federated Edge Learning. Thanks to the requirement of real-time processing of data in many applications and the improvement in computing power and storage capacity of smart devices, research into intelligence at the edge has proliferated in recent years [2]. An example of cross-device federated learning algorithm that has been studied extensively is FedAvg [5], which includes a large network of highly diverse devices participating in learning a global predictive model. On the other hand, one of the early adopters of federated edge learning is [26], who propose the use of GD for a FedAvg-type algorithm. In such distributed computing methods, the cost of communication is generally significant and poses serious problems in operation, especially in networks where limited bandwidth and high latency are prominent. [4] proposes a method to reduce the uplink communication costs; however, this method requires uploading the Hessian matrix to the server for aggregation. [14] proposes FEDL, a first-order method for federated learning and shows that there is always a trade-off between computation and communication: to require less communication during training, the amount of computational processing in edge workers/devices has to increase.

2 DONE: Distributed Approximate Newton-Type Method for Federated Edge Learning

2.1 Optimization Problem

In the context of federated edge learning, there are n edge workers, located at different sites and communicating with an edge aggregator[2] to learn a model w^* which is a solution to the following problem

$$\min_{w\in\mathbb{R}^d} f(w) \doteq \frac{1}{n}\sum_{i=1}^{n} f_i(w), \tag{1}$$

where $f_i(w)$ is the loss function of worker i. Each worker i has a local dataset containing a collection of D_i samples $\{a_j, y_j\}_{j=1}^{D_i}$, where $a_j \in \mathbb{R}^d$ is an input and $y_j \in \mathbb{R}$ can be a target response (or label). Then the (regularized) loss function of each worker i is the average of losses on its data points[3] as follows

2. The roles of edge workers and the aggregator are respectively similar to worker nodes and parameter server in data center-type distributed machine learning.

3. In machine learning, this problem is called the empirical risk minimization (ERM).

$$f_i(w) = \frac{1}{D_i}\sum_{j=1}^{D_i} l(w,(a_j,y_j)) + \frac{\lambda}{2}\|w\|^2, \tag{2}$$

where the regularization term $\frac{\lambda}{2}\|w\|^2$ is added to improve stability and generalization. Throughout this paper, we use $\|\cdot\|$ as the euclidean norm for vectors, and spectral norm for matrices. Some examples of the loss function are: linear regression with $l(w,(a_j,y_j)) = \frac{1}{2}(\langle a_j,w\rangle - y_j)^2, y_j \in \mathbb{R}$, and logistic regression with $l(w,(a_j,y_j)) = \log\left(1+\exp(-y_j\langle a_j,w\rangle)\right), y_j \in \{-1,1\}$, where $\langle x,y\rangle$ denotes the inner product of vectors x and y.

Assumption 1. *The function $f:\mathbb{R}^d \to \mathbb{R}$ is twice continuously differentiable, L-smooth, and λ-strongly convex, $\lambda > 0$, i.e.,*

$$\lambda I \preceq \nabla^2 f(w) \preceq LI, \quad \forall w \in \mathbb{R}^d,$$

where $\nabla^2 f(w) \in \mathbb{R}^{d\times d}$ and $I \in \mathbb{R}^{d\times d}$ denote the Hessian of f at w and identity matrix, respectively.

We use Assumption 1 throughout this paper. We note that strong convexity and smoothness in Assumption 1 can be found in a wide range of loss functions such as linear regression and logistic regression as above. We also denote $\kappa \doteq \frac{L}{\lambda}$ the condition number of f, where large κ means f is ill-conditioned, which may require high computational complexity to optimize.

2.2 Challenges to Distributed Newton's Method

We first review the Newton's method [11] to solve (1) as follows

$$\begin{aligned} w_{t+1} &= w_t - \left(\nabla^2 f(w_t)\right)^{-1}\nabla f(w_t) \\ &= w_t - \Big(\frac{1}{n}\sum_{i=1}^n \nabla^2 f_i(w_t)\Big)^{-1}\Big(\frac{1}{n}\sum_{i=1}^n \nabla f_i(w_t)\Big). \end{aligned} \tag{3}$$

We denote the *global* Newton direction in iteration t as $d_t \doteq -\left(\nabla^2 f(w_t)\right)^{-1}\nabla f(w_t)$. In the vanilla Newton's method, the update rule (3) becomes $w_{t+1} = w_t + d_t$.

One might be tempted to allow each worker to send $\nabla f_i(w_t)$ and $\nabla^2 f_i(w_t)$ to the aggregator for updating (3). However, the crux to designing a distributed Newton's method is *Hessian-free communication* and *inverse-Hessian-free computation*. Indeed, sending Hessians of size $O(d^2)$ over the network or computing the inverse Hessian with complexity $O(\sum_{i=1}^n D_i d^2 + d^3)$, where d is the dimension of data, is considered impractical considering high-dimensional feature vector or "big data" size.

Some approaches such as in [13], [15] allow each worker to approximately calculate $\left(\nabla^2 f_i(w_t)\right)^{-1}\nabla f(w_t)$ and send this vector to the edge aggregator for the following update

$$w_{t+1} = w_t - \frac{1}{n}\sum_{i=1}^n \left(\nabla^2 f_i(w_t)\right)^{-1}\nabla f(w_t).$$

We next review the key ingredient that helps DONE overcome these challenges.

2.3 Richardson Iteration Review

The Newton direction d_t is the solution to the following system of linear equations:

$$\nabla^2 f(w_t) d_t = -\nabla f(w_t).$$

As mentioned earlier, the cost of exactly solving this equation is $O(d^3)$, which can be impractical if the model is high-dimensional. Here we describe the Richardson iteration, a method to solve this approximately but at a lower cost.

The purpose of the Richardson iteration is to find the vector x^* satisfying the linear system $Ax^* = b$, assuming $A \in \mathbb{R}^{d\times d}$ is a symmetric and positive definite matrix.

$$x_k = (I - \alpha A)x_{k-1} + \alpha b, \quad k = 1,2,\ldots$$

Let $\lambda_{max}(A)$ and $\lambda_{min}(A)$ denote the largest and smallest eigenvalues of A, respectively. The Richardson iteration converges, i.e., $\lim_{k\to\infty} x_k = x^* = A^{-1}b$, if and only if

$$0 < \alpha < \frac{2}{\lambda_{max}(A)}, \tag{4}$$

which ensures $\|I - \alpha A\| < 1$. The details of Richardson iteration analysis can be found in [27]. From another viewpoint, the Richardson iteration is equivalent to using the GD method to solve the following quadratic problem

$$\min_{y\in\mathbb{R}^d} \quad \frac{1}{2}\langle y, Ay\rangle - \langle y, b\rangle.$$

2.4 DONE Algorithm

The general idea of DONE, as presented in in Algorithm 1, is each client finds its local Newton direction vector, and the server will aggregate these local vectors to approximate the global Newton directions. We see that $2T$ is the number of communication iterations between the aggregator and edge workers. Note that the exact global gradient $\nabla f(w_t)$ is required (line 5), which is the key to the convergence of DONE to be shown later. Define the *local* Newton direction on client i in iteration t as $d_{i,t} \doteq \left(\nabla^2 f_i(w_t)\right)^{-1}\nabla f_i(w_t)$. To find the local Newton direction $d_{i,t}$, Algorithm 1 uses R Richardson iterations, resulting in the *approximate* local Newton direction $d_{i,t}^R$ (line 8).

The gist of DONE . Even though each worker uses the Richardson iteration to obtain its local Newton direction $d_{i,t}^R$, their average d_t^R at the aggregator well approximates the global Newton direction when $\alpha \le 1/R$. This is demonstrated in the following result.

Theorem 1. *Consider symmetric and positive definite matrices $A_i \in \mathbb{R}^{d\times d}, i = 1,\ldots,n$. Let $A = \frac{1}{n}\sum_{i=1}^n A_i$, and x_k and $x_{i,k}$ follow the Richardson iteration as follows*

$$\begin{aligned} x_k &= (I - \alpha A)x_{k-1} + \alpha b \\ x_{i,k} &= (I - \alpha A_i)x_{i,k-1} + \alpha b, \quad k = 1,2,\ldots \end{aligned}$$

with $0 < \alpha < \frac{2}{\lambda_{max}(A)}$, then we have

$$\lim_{k\to\infty} x_k = A^{-1}b = x^*, \quad \lim_{k\to\infty} x_{i,k} = A_i^{-1}b.$$

Especially, when $x_{i,0} = x_0$ and $\alpha \le \min\{\frac{1}{k}, \frac{1}{\hat{\lambda}_{max}}\}$, where $\hat{\lambda}_{max} \doteq \max_i \lambda_{max}(A_i)$, we have

$$\left\|\frac{1}{n}\sum_{i=1}^n x_{i,k} - x^*\right\| \le E_1 + E_2,$$

where

$$E_1 = \|x_k - x^*\| \leq \|(I - \alpha A)^k\| \|x_0 - x^*\|,$$
$$E_2 = \left\| \frac{1}{n}\sum_{i=1}^{n} x_{i,k} - x_k \right\| \leq O\bigg(\frac{\nu}{k^2}\big(\|b\| + \|x_0\|\big)\bigg),$$
$$\nu = \left\| A^2 - \frac{1}{n}\sum_{i=1}^{n} A_i^2 \right\|.$$

The proof to Theorem 1 can be found in 6.1. Observing that $d_{i,t}^r, \nabla^2 f_i(w_t)$, and $-\nabla f(w_t)$ of DONE play the same roles as $x_{i,k}, A_i$, and b in the above theorem, respectively, we obtain the following remarks.

- The only parameter of DONE to be fine-tuned is α. As it is impractical to compute the eigenvalues of $\nabla^2 f_i(w_t)$ to guarantee convergence as in (4) , the Richardson iteration may not converge. In practice, however, a sufficiently small α almost always works, but very small α can lead to slow convergence (c.f. Section 4).
- The theory of DONE requires full local data passing and full participation of all edge workers. However, using mini-batches is common in machine learning to reduce the computation bottleneck at the worker. Another critical issue is the straggler effect, in which the run-time in each iteration is limited by the slowest worker (the straggler) because heterogeneous edge workers compute and communicate at different speeds. Thus, choosing a subset of participating workers in the aggregation phase is a practical approach to reduce the straggler effect. In Section 4, we will experimentally show that DONE works well with mini-batches and worker subset sampling[4] when approximating the Hessians $\nabla^2 f_i(w_t)$ and mitigating the straggler effect, respectively.
- It is obvious that the larger R and T, the higher computation and communication complexities of DONE, yet the better accuracy and smaller optimal gap, for the Newton direction and convergence, respectively. In the next section, we will quantify how much R and T affect the convergence of DONE.
- In Theorem 1, the error of the approximation is quantified by two terms, E_1 and E_2. E_1 is the error between the *centralized* Richardson iterate and the solution x^*; in the context of DONE, this is the distance between the approximate Newton direction (if all Hessians are sent to the server) and the true Newton direction. On the other hand, E_2 is the error between the *centralized* iterate and the average of *decentralized* iterates. Therefore, this distributed approximation incurs an additional error term E_2. Also, E_2 depends primarily on $\nu = \|A^2 - \frac{1}{n}\sum_{i=1}^{n} A_i^2\|$, which quantifies the heterogeneity in edges' data. In other words, if statistical heterogeneity is more significant, each edge must run more Richardson iterations to approximate the Newton direction to the same precision. Finally, we observe a relationship between α and k (or R in the case of DONE) here: to distributively approximate the Newton direction, α must decrease as k (or R) increases to avoid divergence of the average Richardson iterate.

4. By uniformly at random choosing a subset of B data points from D_i samples and a subset of $S \leq n$ workers, respectively.

Algorithm 1. DONE

```
1:  input: $T, R, \alpha, w_0$
2:  for $t = 0$ to $T-1$ do
3:    Aggregator sends $w_t$ to all edge workers
4:    for all edge workers $i = 1, \ldots, n$ in parallel do
5:      Send $\nabla f_i(w_t)$ to the edge aggregator, then receives the
        aggregated gradient $\nabla f(w_t) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_t)$ sent
        back by the edge aggregator.
6:      Set $d_{i,t}^0 = 0$
7:      for $r = 1$ to $R$ do
8:        $d_{i,t}^r = \big(I - \alpha\nabla^2 f_i(w_t)\big)d_{i,t}^{r-1} - \alpha\nabla f(w_t)$
9:      end for
10:     Send $d_{i,t}^R$ to the aggregator
11:   end for
12:   Aggregator receives $d_{i,t}^R$ from all workers and updates:
        $w_{t+1} = w_t + \eta_t d_t^R, \text{where } d_t^R \doteq \frac{1}{n}\sum_{i=1}^{n} d_{i,t}^R.$   (5)
13: end for
```

3 Convergence and Complexity Analysis

In this section, we will provide the convergence and complexity analysis of DONE, and compare it with distributed GD, DANE [13], and FEDL [14].

3.1 Convergence Analysis

We see that each edge worker has R computation rounds, and the complexity at each round is calculating $d_{i,t}^r$ on line 8. It is straightforward to see that the bottleneck of this step is performing the matrix-vector product with $O(D_i \cdot d^2)$ computation complexity. However, this complexity can be reduced to $O(D_i \cdot d)$ with a special class of generalized linear models (GLM) [28] having the linear term $\langle a_j, w\rangle$ in its loss function, e.g., regularized linear regression or logistic regression. Indeed, from (2), the Hessian of f_i in these learning tasks is

$$\nabla^2 f_i(w_t) = \frac{1}{D_i}\sum_{j=1}^{D_i} \beta a_j a_j^T + \lambda I,$$

where β is a scalar that depends on $\langle a_j, w_t\rangle$. Therefore, the matrix-vector product $\nabla^2 f_i(w_t) d_{i,t}^{r-1}$ becomes

$$\frac{1}{D_i}\sum_{j=1}^{D_i} \beta a_j \langle a_j, d_{i,t}^{r-1}\rangle,$$

which contains only vector-vector products with computation complexity $O(D_i \cdot d)$. Therefore, with GLM, the computation complexity of each worker i using DONE is $O(D_i \cdot d \cdot R \cdot T)$, which only scales linearly with respect to (w.r.t.) the data size and feature dimension.

Before analyzing the effects of R and T on DONE's convergence, we need the following standard assumption for the Newton's method analysis [11].

Assumption 2. *The Hessian of f is M-Lipschitz continuous, i.e., $\|\nabla^2 f(w) - \nabla^2 f(w')\| \leq M\|w - w'\|, \forall w, w' \in \mathbb{R}^d$.*

Here, the value of M measures how well f can be approximated by a quadratic function, e.g., $M = 0$ for when f is quadratic such as in the linear regression case. In this work, we focus on the global convergence of DONE without using the backtracking line search. In order to enable the global convergence of the Newton-type method, we choose an adaptive stepsize of η_t introduced in [29]:

$$\eta_t = \min\left\{1,\ \frac{\lambda^2}{L\|\nabla f(w_t)\|}\right\}. \quad (6)$$

In the damped Newton phase, the edge aggregator will update $w_{t+1} = w_t + \eta_t d_t^R$ with an adaptive step size $\eta_t = \frac{\lambda^2}{L\|\nabla f(w_t)\|}$ to ensure d_t^R is a descent direction. When the damped Newton phase finishes, $\eta_t = 1$ and the algorithm enters the pure Newton phase. For practical DONE, we need to choose a finite value for R, which means that d_t^R is an approximation of the Newton direction. We next define a parameter δ that measures how well d_t^R approximates the true Newton direction. In the sequel, for the ease of presentation, we denote $g_t \doteq \nabla f(w_t)$, $H_t \doteq \nabla^2 f(w_t)$, and $\hat{d}_t \doteq -H_t^{-1} g_t$.

Definition 1. *d_t^R is called a δ-approximate $\hat{d}_t$ if*

$$\|\hat{d}_t - d_t^R\| \leq \delta\|\hat{d}_t\|.$$

In other words, δ captures the inexact level of the approximate solution d_t^R to the equation $H_t x^* = -g_t$ with the true solution $x^* = \hat{d}_t$.

The following result shows the relationship between R and δ, and how the δ-approximation affects the convergence rate of DONE.

Lemma 1. *Let Assumptions 1 and 2 hold. For an arbitrary small $\alpha \leq \min\{\frac{1}{R}, \frac{1}{\lambda_{max}}\}$, if we set*

$$\delta = \|(I - \alpha A)^R\| + O\Big(\frac{\nu L}{R^2}\Big),$$
$$t_0 = \max\left\{0, \left\lceil \frac{2L}{\lambda^2\|\nabla f(w_0)\|} \right\rceil - 2\right\},$$
$$\gamma = \frac{L}{2\lambda^2}\|\nabla f(w_0)\| - \frac{t_0}{4} \in \Big[0, \frac{1}{2}\Big);$$

then DONE has

$$\|w_t - w^*\| \leq \begin{cases} \frac{1}{\kappa}(t_0 - t + \frac{2\gamma}{1-\gamma}) + \frac{\delta}{\kappa}, t \leq t_0 \\ \frac{2\gamma^{2^{t-t_0}}}{\kappa(1-\gamma^{2^{t-t_0}})} + (\delta\kappa)^t\|w_0 - w^*\|, t > t_0. \end{cases} \quad (7)$$

The proof can be found in 6.2. Lemma 1 shows that in damped phase, $\|w_t - w^*\|$ is decreased by at least a constant at each iteration. In pure Newton phase, DONE has a linear-quadratic convergence rate. The quadratic term in this lemma is equivalent to [29], whereas the linear term is due to the δ-approximation of Newton direction of DONE. The δ-approximation includes two terms. The first term comes from using Richarsion Interaction to approximate true Newton direction and the second one depends on the heterogeneity in edges' data. It is obvious that when R increases, δ decreases, and thus the linear term disappears, recovering the quadratic convergence of the adaptive Newton method. Lemma 1 also shows when κ is large, i.e., more computation rounds are needed to reduce the approximation error δ to guarantee $\delta\kappa < 1$.

3.2 Complexity Analysis

We next address the communication complexity $O(T)$ for the global convergence of DONE to w^*.

Definition 2. *We define w_T an ϵ-optimal solution to (1) if*

$$\|w_T - w^*\| \leq \epsilon.$$

From Lemma 1, we see that there are regimes in which DONE has linear or quadratic convergence rate, and these regimes depend on the δ setting and the initialization w_0 in a neighborhood of w^*. In the following we will identify two extreme regimes.

In the first regime where the δ-approximation error dominates the bound in (7), i.e., $\|w_0 - w^*\| \geq \frac{2t\gamma^{2^{t-t_0}}}{\kappa(\delta\kappa)^t(1-\gamma^{2^{t-t_0}})}$ then DONE has linear convergence.

Theorem 2. *If $\|w_0 - w^*\| \geq \frac{2t\gamma^{2^{t-t_0}}}{\kappa(\delta\kappa)^t(1-\gamma^{2^{t-t_0}})}$, and assume that $\delta\kappa < 1$, then DONE has linear convergence*

$$\|w_t - w^*\| \leq 2(\delta\kappa)^t\|w_0 - w^*\|,$$

and its communication complexity is

$$T = O\Big(\delta\kappa \log\frac{1}{\epsilon}\Big). \quad (8)$$

The proof can be found in 6.3. In the second regime where the δ-approximation error is negligible, i.e., $\|w_0 - w^*\| < \frac{2t\gamma^{2^{t-t_0}}}{\kappa(\delta\kappa)^t(1-\gamma^{2^{t-t_0}})}$, then DONE has quadratic convergence.

Theorem 3. *If $\|w_0 - w^*\| < \frac{2t\gamma^{2^{t-t_0}}}{\kappa(\delta\kappa)^t(1-\gamma^{2^{t-t_0}})}$ then DONE has quadratic convergence*

$$\|w_t - w^*\| \leq \frac{4t\gamma^{2^{t-t_0}}}{\kappa(1 - \gamma^{2^{t-t_0}})},$$

and its communication complexity is

$$T = O\Big(\log\log\frac{1}{\epsilon}\Big). \quad (9)$$

The proof can be found in 6.4. We see that controlling δ gives a trade-off between computation and communication complexities, where very small δ increases the computation iterations but may significantly reduce the communication cost with quadratic convergence, and vice versa.

3.3 Comparison With Other Distributed Methods

If a first-order method such as GD is used for federated edge learning, the edge aggregator first receives $\nabla f_i(w_t), \forall i$, updates

$$w_{t+1} = w_t - \eta\Big(\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(w_t)\Big), \tag{10}$$

and then sends w_{t+1} to all workers for next iteration. By choosing $\eta = \frac{2}{\lambda+L}$, it has been shown that the communication complexity of GD for ϵ-optimal solution [12, Theorem 2.1.5] is $\kappa\log\frac{1}{\epsilon}$. We note that this distributed GD (10) is a common approach in data center-type distributed machine learning [30], [31], which is different from FedAvg-type GD in [5], [26]. While FedAvg-type GD (or SGD) allows each device/worker to update its local model multiple times using local gradient information (thus its alternative name is Local GD/SGD [32]), the update in (10) requires each client/worker to calculate its gradient once and send to the aggregator. Compared to first-order methods, without multiple local model updates, DONE is more in line with distributed GD in (10) than FedAvg-type GD algorithms.

Regarding other related Newton-type methods, DiSCO [25] requires several communication rounds between the aggregator and workers even for one Newton direction update, whereas GIANT [15] requires homogenous data, in which data must be collected centrally, shuffed, and distributed evenly to each node. Thus, both DiSCO and GIANT are only applicable to data center-type distributed machine learning. On the other hand, DANE [13] and FEDL [14], approximate Newton-type methods, are comparable to DONE have been shown to be a good candidate for federated learning. In DANE and FEDL, two communication rounds are used in one global iteration. In the first round, the workers compute the local gradients $\nabla f_i(w_t)$, which are then aggregated into the global gradient $\nabla f(w_t)$. In the second round, each worker solves a special local optimization using $f_i(w)$ and $\nabla f(w_t)$. It has been shown that to achieve an ϵ-accurate solution,[5] DANE requires $O\Big(\frac{\kappa^2}{D_i}\log(dn)\log\frac{1}{\epsilon}\Big)$ iterations while FEDL requires $O\Big(\frac{1}{\Theta}\log\frac{1}{\epsilon}\Big)$ with $\Theta \in (0,1)$. A comparison of communication and computation complexities of DONE, distributed GD, FEDL, and DANE for a ridge regression task (quadratic loss) is summarized in Table 1.

4 Experimental Results and Discussion

In this section, we evaluate the performance of DONE when the data sources across edge workers are non-i.i.d. and heterogeneous. We first show the effect of various hyperparameters and how to choose those parameters for different scenariosc. We then compare DONE with the Newton's method (3), FEDL, DANE, GIANT, and GD (10) using real datasets. The experimental outcomes show that DONE has the similar performance to the Newton's method with the same hyperparameters settings and achieves the performance improvement over FEDL, DANE, and GD in terms of testing accuracy, running time, and communication cost when all algorithms have a same target accuracy.

4.1 Experimental Settings

We implement several classification and regression experiments. For classification tasks, to compare the performance of DONE with other methods, we use three real datasets including MNIST, FEMNIST, and Human Activity Recognition generated in federated settings. We also use a synthetic dataset for a linear regression task, to monitor the effect of the condition number κ. We distribute the complete data to $n = 32$ edge workers ($n = 30$ only for Human Activity) and randomly split the data into two parts: 75% for training and 25% for validation. The detail of all datasets are provided as follow:

Synthetic: To generate non-i.i.d. data, each egde worker i has a collection of D_i samples $\{a_j, y_j\}_{j=1}^{D_i}$ following the linear regression model $y_j = \langle w^*, a_j + c_j\rangle$ with w^*, $a_j \in \mathbb{R}^d$, $c_j \in \mathbb{R}$, $a_j \sim \mathcal{N}(0, \sigma_j\Sigma)$, where $\sigma_j \sim \mathcal{U}(1,30)$, $c_j \sim \mathcal{N}(0,1)$, and $\Sigma \in \mathbb{R}^{d\times d}$ is a diagonal covariance matrix with $\Sigma_{i,i} = i^{-\tau}, i = 1,\dots,d$. The main purpose of synthetic data is to control the condition number κ. By setting $\tau = \frac{\log(\kappa)}{\log(d)}$, $\kappa = d^{\tau}$ is the ratio between the maximum and minimum covariance values of Σ. To model a heterogeneous setting, each edge has a different data size in the range $[540, 5630]$.

MNIST [33]: A handwritten digit dataset including 70,000 samples and 10 labels. In order to simulate a heterogeneous and non-i.i.d. data setting, each worker has only 3 labels and the data sizes vary in the range $[219, 3536]$.

FEMNIST: A dataset partitioned from Extended MNIST [34] which includes 62-class digit following [35] . To generate federated setting, only 10 lower case characters ('a'-'j') are selected and distributed 32 egdes (5 classes per edge).

Human Activity Recognition [36]: A dataset collected from mobile phone accelerometers and gyroscope of 30 individuals, each performing one of six different activities: sitting, walking, walking upstairs, walking downstairs, lying down and standing. This dataset naturally captures the federated non-i.i.d. and heterogeneous characteristics. By considering each individual as an edge, we have 30 edges in total and the data size of each edge is in the range $[281, 383]$.

In our experiment, the regression task on the synthetic dataset uses a linear regression model with mean squared error loss, and the classification task on real datasets uses multinomial logistic regression (MLR) models with cross-entropy loss. We implement all algorithms using PyTorch version 1.8.0 and evaluate on Tesla K80 GPU. Each experiment is run 10 times for statistical reports.

4.2 Effect of κ

We first verify our theoretical findings by observing the convergence behavior of DONE on a wide range values for κ, including

5. The definition of ϵ-accuracy in [13], [14] is $f(w_t) - f(w^*) \le \epsilon$.

TABLE 1
Communication and Computation Complexity Comparison for the Ridge Regression Problem

Method	Communication	Computation
DONE	$O\Big(\delta\kappa\log\frac{1}{\epsilon}\Big)$	$O\Big(D_i\cdot d\cdot R\cdot\delta\kappa\log\frac{1}{\epsilon}\Big)$
	$O\Big(\log\log\frac{1}{\epsilon}\Big)$	$O\Big(D_i\cdot d\cdot R\cdot\log\log\frac{1}{\epsilon}\Big)$
DANE	$O\Big(\frac{\kappa^2}{D_i}\log(dn)\log\frac{1}{\epsilon}\Big)$	$O\Big(d\cdot R\cdot\kappa^2\log(dn)\log\frac{1}{\epsilon}\Big)$
FEDL	$O\Big(\frac{1}{\Theta}\log\frac{1}{\epsilon}\Big)$	$O\Big(D_i\cdot d\cdot R\cdot\frac{1}{\Theta}\log\frac{1}{\epsilon}\Big)$
GD	$O\Big(\kappa\log\frac{1}{\epsilon}\Big)$	$O\Big(D_i\cdot d\cdot\kappa\log\frac{1}{\epsilon}\Big)$

DONE achieves a global linear or quadratic convergence depending on the regime specified in Theorems 2 and 3. The computation complexity of DONE *with GLM is comparable to that of the first-order methods. GIANT is not considered for the comparison as it only has a local convergence rate.*

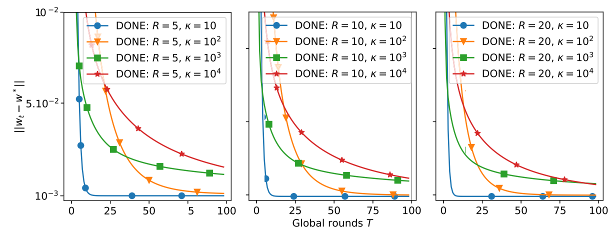


Fig. 1. Effects of κ on convergence of DONE ($\alpha = 0.05, R = 5, 10, 20$) on regression task. Larger κ slows down the convergence of DONE; therefore, DONE requires more local update R to reduce the approximate error δ.

small ($\kappa = 10$), medium ($\kappa \in \{10^2, 10^3\}$), and large ($\kappa = 10^4$) on the synthetic dataset in Fig 1. Using the linear regression model, we can obtain $\lambda_{i,max}$, the largest eigenvalues of the Hessian of f_i of each worker i. We then choose $\alpha \leq \frac{1}{\lambda_{max}}$, e.g., $\alpha = 0.05$ and $\alpha \leq \frac{1}{R}$ following Theorem 1. It can be seen that using choosen values of α and R allows DONE to converge approximately to the solution with all settings of κ. However, with given the same error tolerance, larger κ requires larger R to reduce the approximation error δ, which is verified in Lemma 1.

4.3 Effect of Hyper-Parameters: α, R

We show the impact of wide ranges of α, R on the convergence of DONE for MNIST, FEMNIST, and Human Activity in Figs. 2, 3, and 4, respectively. To monitor the effect of α, we fix the value of R and vice versa. The results demonstrate that there exist sets of sufficiently small α and large R following the condition such that $\alpha < \frac{1}{R}$ allowing DONE to converge. We observe that both larger α and R speed up the convergence of DONE as the larger R allows DONE to approximate true Newton direction closely. However, increasing R comes at a cost of higher local computation, and increasing α ($\alpha \geq 0.035$ for MNIST, $\alpha \geq 0.015$ for FEMNIST, and $\alpha \geq 0.03$ for Human Activity) can lead to divergence of DONE. Both R and α should be tuned carefully depends on the heterogeneity of data. The more heterogeneous data is, the less value of α and the higher value of R is considered to reduce the approximate term δ. In federated edge computing where each edge worker has a powerful computational capacity, it is reasonable to use larger R and smaller α to handle the heterogeneity and also to reduce the cost of communication.

4.4 Effect of Mini-Batch Sampling

Even though our analysis of DONE requires computing full-batch Hessian-gradient products, we also consider an additional case where we sample a mini-batch of size B (where

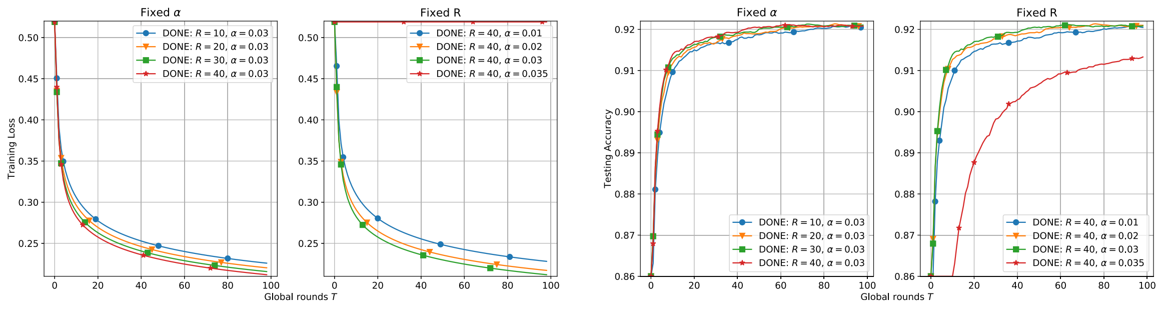


Fig. 2. Effects of various values of α and R on MNIST.

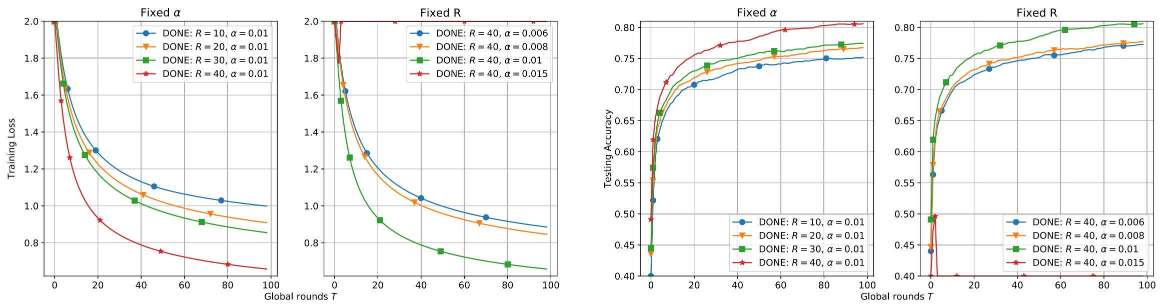


Fig. 3. Effects of various values of α and R on FEMNIST.

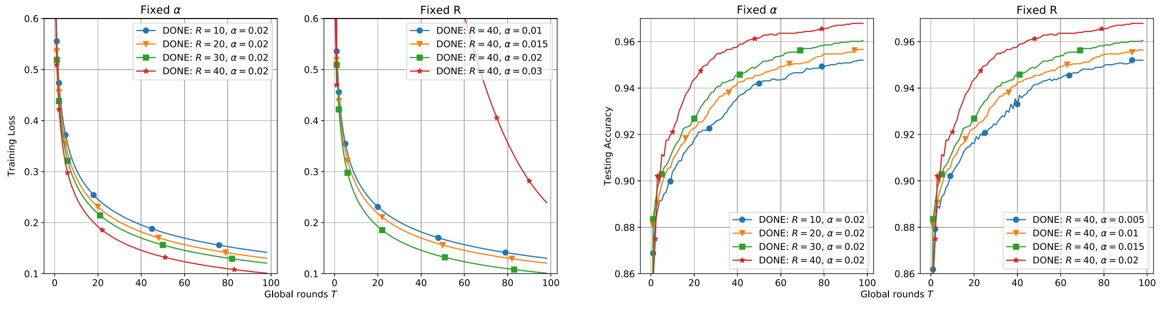
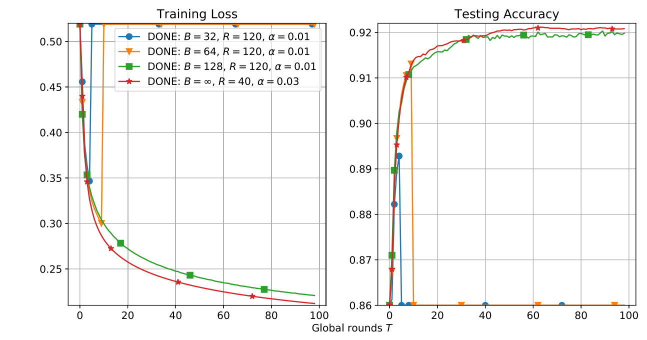
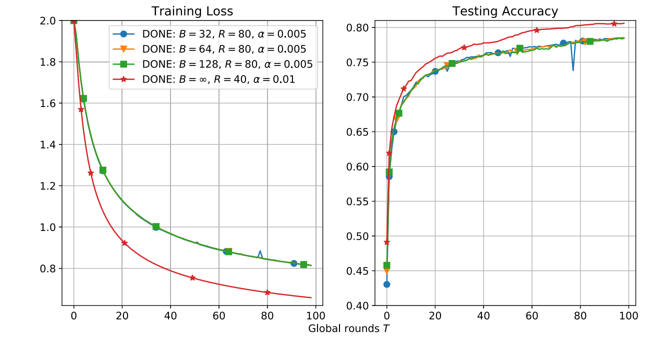


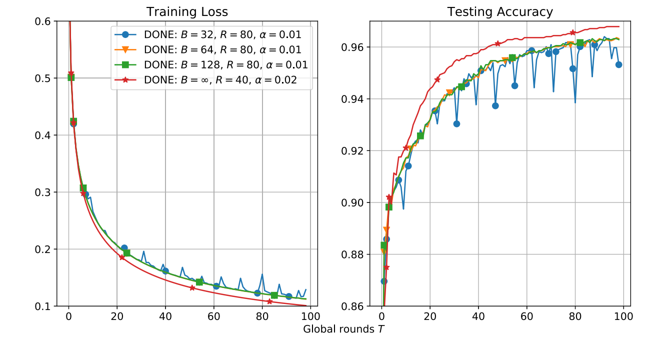
Fig. 4. Effects of various values of α and R on human activity.



(a) MNIST

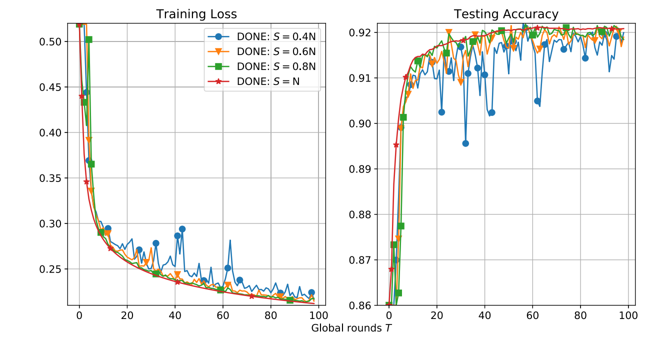


(b) FEMNIST

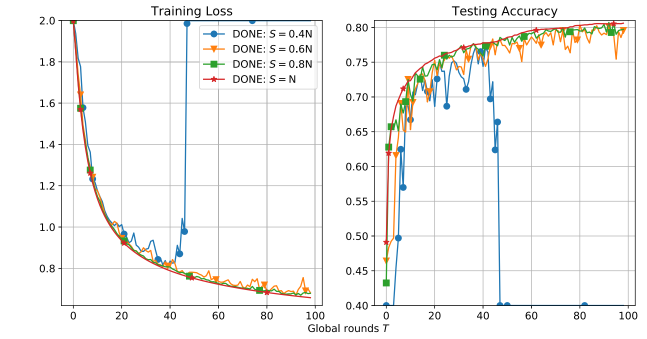


(c) Human Activity.

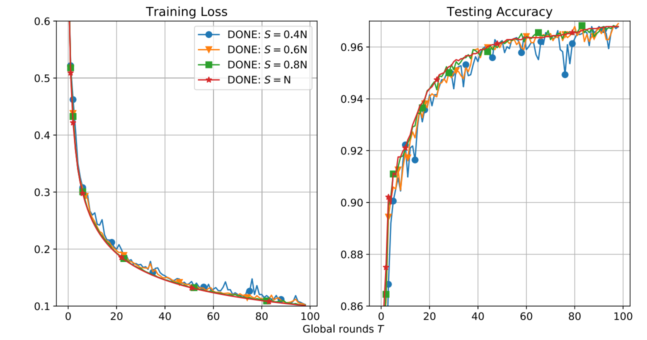
Fig. 5. Effects of mini-batch sampling. DONE are more stable with the larger batch size.



(a) MNIST



(b) FEMNIST



(c) Human Activity.

Fig. 6. Effect of sampling a subset of edge workers.

$B \in \{32, 64, 128\}$) in each round to approximate the true Hessian-gradient product. From the experimental results, using a mini-batch requires a smaller value of α than that of full-batch. In Figs. 5a, 5b, and 5c, by reducing the value of α and increasing the value of R correspondingly, the performance of DONE using mini-batches is close to that of DONE using full batches. In addition, using small mini-batches can lead to instability of DONE: e.g., when $B = 32$ DONE diverges in case of MNIST and becomes less stable in the cases of FEMNIST and Human Activity.

4.5 Effect of Edge Worker Sampling

In practice, besides using mini-batches to reduce computation, it is critical to address the straggler's effect. We consider a scenario when a subset of workers of size S is selected randomly for aggregation. We keep using the same experimental setting as above but randomly select the value of S in $\{N, 0.8N, 0.6N, 0.4N\}$. In Figs. 6a, 6b, and 6c, DONE converges in all choices of $S \geq 0.6N$. As expected, larger S allows DONE to converge faster and be more stable. On the other hand, when $S \leq 0.4N$ we observe the deterioration of DONE's performance, especially in the case of FEMNIST.

4.6 Performance Comparison With Distributed Algorithms

We finally compare DONE with the Newton's method, GD, DANE, and FEDL. For a fair comparison, we fix a same number of communication rounds (T) and the number of local updates (R) for all algorithms. We then use grid search to fine-tune the hyper-parameters w.r.t. the highest test accuracy and stability of each algorithm. We fix $\eta = 1$ for DANE and choose the best regularization parameter μ in $\{0, \lambda, 3\lambda\}$ for DANE and GIANT [6]. We follow the setting of [14] for FEDL. We also apply the Richardson iteration to the true Newton's method (3) since it is impractical to do the inverse Hessian. Newton's method with Richardson

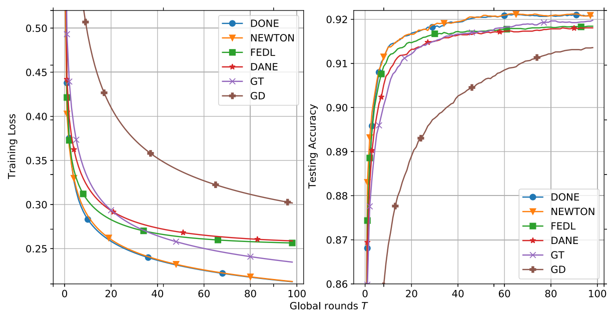
6. We direct the reader to [13] for the meaning of DANE's parameters.

TABLE 2
Performance Comparison on Three Real Datasets. We fix $R = 40, T = 100$ and for All Algorithms

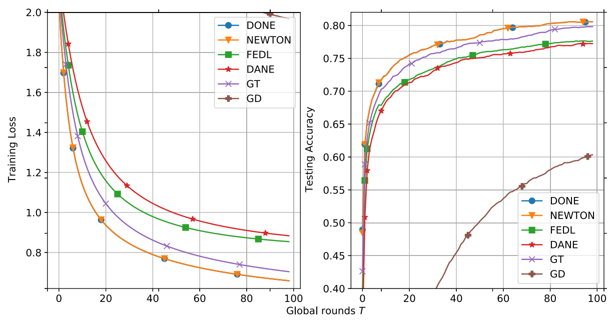
Dataset	Algorithm	α (γ)	Accuracy	Running Time (ms)
MNIST	DONE	0.03	$\mathbf{92.11} \pm 0.01$	399.00 ± 2.52
	FEDL	0.04	91.89 ± 0.01	660.27 ± 2.87
	DANE	0.04	91.84 ± 0.01	742.22 ± 3.11
	GIANT		91.89 ± 0.01	464.52 ± 2.41
	Newton	0.03	$\mathbf{92.11} \pm 0.01$	1176.60 ± 3.57
	GD	0.2	91.35 ± 0.02	42.79 ± 3.09
FE-MNIST	DONE	0.01	$\mathbf{80.60} \pm 0.02$	372.97 ± 2.16
	FEDL	0.01	78.28 ± 0.01	486.08 ± 2.11
	DANE	0.01	77.57 ± 0.01	490.28 ± 2.25
	GIANT		79.61 ± 0.01	424.25 ± 1.16
	Newton	0.01	$\mathbf{80.60} \pm 0.02$	911.26 ± 3.16
	GD	0.02	60.58 ± 0.03	39.60 ± 2.68
Human Activity	DONE	0.02	96.78 ± 0.01	216.42 ± 1.35
	FEDL	0.05	95.90 ± 0.01	246.13 ± 1.49
	DANE	0.05	95.82 ± 0.01	256.79 ± 1.53
	GIANT		96.13 ± 0.02	354.33 ± 1.67
	Newton	0.02	96.78 ± 0.01	583.83 ± 2.71
	Newton	0.03	$\mathbf{96.90} \pm 0.01$	
	GD	0.1	80.02 ± 0.02	23.84 ± 1.95

γ is the learning rate of DANE, FEDL, and GD.

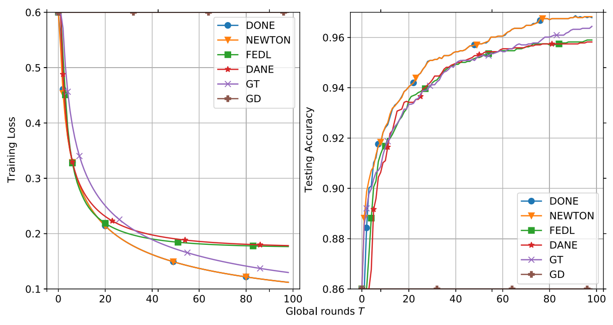
iteration requires all edge workers to send the local Newton direction to the server for aggegation at each single local update, hence it actually takes $R.T$ communication rounds. In comparison with Newton's method, two separate cases are considered: when the Newton's method has the same values of hyper-parameter α with DONE and uses fine-tuned α. We compare the accuracy and running time in Table 2 and the convergence of all algorithms in Figs. 7a, 7b, and 7c. The experimental results show that DONE has a similar performance to Newton's method on the same set of α and R. As highlighted in Theorem 1, when α is small and R is large enough, the Richardson iteration allows DONE to catch up with the Newton's method. There is only a small gap (0.12%) when Newton's method has a larger value of α which can be seen when using the Human Activity, a highly non-i.i.d dataset. To obtain similar performance with Newton's method in this scenario, DONE needs to run more local iterations ($R = 50$). In the case of FEMNIST and MNIST, DONE and the Newton's method have the same value of fine-tuned α. In comparison with others, DONE improves from FEDL, DANE, GAINT, and GD in all real datasets. For MNIST, the improvement in test accuracy compared to FEDL, DANE, GIANT, and GD are approximately 0.22%, 0.27%, 0.22%, and 0.76%, respectively. The corresponding figures are 2.32%, 3.03%, 0.99%, and 20.02% for FEMNIST, and 0.88%, 0.96%, 0.65%, and 16.76% for Human Activity. Finally, to compare the communication and computation complexities among all algorithms, we first fix $T = 100$ and compare the running time of all algorithms. The results in Table 2 show that DONE has the smallest running time. Additionally, in Table 3, we set the same target accuracy and compare the number of global rounds T needed by each algorithm to achieve that accuracy. Overall, DONE shows a marked improvement from GIANT, DANE, and FEDL, requiring much fewer iterations to achieve the same accuracy.



(a) MNIST



(b) FEMNIST



(c) Human Activity.

Fig. 7. Convergence comparisons. We fix $R = 40, T = 100$ for all algorithms and use same values of α and γ in Table 2.

TABLE 3
Communication Times T for All Algorithms to Achieve the Same Target Accuracy

Dataset	Algorithm				Target Accuracy
	DONE	GIANT	FEDL	DANE	
MNIST	**28**	59	70	100	**91.84**
FEMNIST	**31**	62	75	100	**77.57**
Human Activity	**55**	77	86	100	**95.82**

5 Conclusion

In this work, we develop a distributed approximate Newton-type algorithm (DONE) suitable for federated edge learning. We show that DONE effectively approximates the true Newton direction using the Richardson iteration when the loss functions are strongly convex and smooth. Additionally, we specify that DONE has global linear-quadratic convergence and provide its computation and communication complexity analysis. Finally, we experimentally verify our theoretical findings and demonstrate the competitiveness of our approach when compared with the distributed GD method, FEDL, and DANE, approximate distributed Newton-type algorithms.

6 Proofs

We provide proofs for the theorems and lemmas.

6.1 Proof of Theorem 1

By Richardson iteration convergence, it is straightforward to have: $\lim_{k\to\infty} x_k = A^{-1}b = x^*$ and

$$\|x_k - x^*\| \leq \|(I-\alpha A)^k\|\|x_0 - x^*\|. \quad (11)$$

Using the Richardson iteration, expanding x_k recursively gives

$$\begin{aligned} x_k &= (I-\alpha A)x_{k-1} + \alpha b \\ &= (I-\alpha A)^k x_0 + \sum_{j=0}^{k-1}(I-\alpha A)^j \alpha b. \quad (12)\end{aligned}$$

Similarly, expanding $x_{i,k}$ recursively gives

$$x_{i,k} = (I-\alpha A_i)^k x_{i,0} + \sum_{j=0}^{k-1}(I-\alpha A_i)^j \alpha b. \quad (13)$$

Taking the average of all $x_{i,k}$ gives

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^n x_{i,k} &= \frac{1}{n}\sum_{i=1}^n\Big((I-\alpha A_i)^k x_{i,0} + \sum_{j=0}^{k-1}(I-\alpha A_i)^j\alpha b\Big) \\ &= \frac{1}{n}\sum_{i=1}^n (I-\alpha A_i)^k x_{i,0} + \frac{1}{n}\sum_{i=1}^n\sum_{j=0}^{k-1}(I-\alpha A_i)^j\alpha b. \quad (14)\end{aligned}$$

Using the Taylor expansion on $(I-\alpha A)^j$ gives

$$\begin{aligned}(I-\alpha A)^j &= \binom{j}{0}I + \binom{j}{1}(-\alpha A)^1 + \binom{j}{2}(-\alpha A)^2 + \ldots \\ &= I - \alpha jA + \alpha^2\frac{j(j-1)}{2}A^2 + \ldots \quad (15)\end{aligned}$$

Taking the sum of this term from $j = 0$ to $k-1$ gives

$$\begin{aligned}\sum_{j=0}^{k-1}(I-\alpha A)^j &= kI - \alpha\frac{k(k-1)}{2}A \\ &\quad + \alpha^2\frac{k(k-1)(k-2)}{3!}A^2 + \ldots \quad (16)\end{aligned}$$

For any matrix A and constant α satisfying $\|\alpha A_i\| < 1, \forall i$, we can respectively express (15) and (16) as

$$(I-\alpha A)^j = I - \alpha jA + O(\alpha^2A^2) \quad (17)$$

$$\sum_{j=0}^{k-1}(I-\alpha A)^j = kI - \alpha\frac{k(k-1)}{2}A + O(\alpha^2kA^2). \quad (18)$$

Substituting (17) and (18) into (12) gives

$$\begin{aligned}x_k &= v(I-\alpha kA + O(\alpha^2A^2))x_0 \\ &\quad + \Big(kI - \frac{k(k-1)}{2}\alpha A + O(\alpha^2kA^2)\Big)\alpha b,\end{aligned}$$

Similarly, substituting (17) and (18) into (13) gives

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^n x_{i,k} &= \frac{1}{n}\sum_{i=1}^n\big(I-\alpha kA_i + O(\alpha^2A_i^2)\big)x_{i,0} \\ &\quad + \frac{1}{n}\sum_{i=1}^n\Big(kI - \alpha\frac{k(k-1)}{2}A_i\Big)\alpha b \\ &\quad + \frac{1}{n}\sum_{i=1}^n O(\alpha^2kA_i^2)\alpha b.\end{aligned}$$

If $A = \frac{1}{n}\sum_{i=1}^n A_i$, $\nu = \big\|A^2 - \frac{1}{n}\sum_{i=1}^n A_i^2\big\|$, and $x_0 = x_{i,0}$ we can bound the distance between x_k and $\frac{1}{n}\sum_{i=1}^n x_{i,k}$ as

$$\Big\|x_k - \frac{1}{n}\sum_{i=1}^n x_{i,k}\Big\| \leq O\Big(\alpha^2\nu\|x_0\|\Big) + O\Big(\alpha^3k\nu\|b\|\Big). \quad (19)$$

Further, if $\alpha \leq \frac{1}{k}$, then

$$\Big\|x_k - \frac{1}{n}\sum_{i=1}^n x_{i,k}\Big\| \leq O\Big(\frac{\nu}{k^2}\big(\|b\| + \|x_0\|\big)\Big). \quad (20)$$

We have

$$\Big\|\frac{1}{n}\sum_{i=1}^n x_{i,k} - x^*\Big\| \leq \Big\|\frac{1}{n}\sum_{i=1}^n x_{i,k} - x_k\Big\| + \|x_k - x^*\| \quad (21)$$

$$\begin{aligned}&\leq O\Big(\frac{\nu}{k^2}\big(\|b\| + \|x_0\|\big)\Big) \\ &\quad + \|(I-\alpha A)^k\|\|x_0 - x^*\|, \quad (22)\end{aligned}$$

where (21) results from the triangle inequality, and (22) is derived from (20) and (11). In addition to the error by centralized Richardson iteration, the distributed version incurs an error from (20).

6.2 Proof of Lemma 1

Using the triangle inequality, we have

$$\begin{aligned}\|w_t - w^*\| &= \|w_{t-1} - w^* + \eta_{t-1}d_{t-1}^R\| \\ &\leq \underbrace{\|w_{t-1} - w^* + \eta_{t-1}\hat{d}_{t-1}\|}_{T_1} + \eta_{t-1}\underbrace{\|d_{t-1}^R - \hat{d}_{t-1}\|}_{T_2}.\end{aligned}$$

According to Theorem 4.1 in [29], let:

$$t_0 = \max\Big\{0, \Big\lceil\frac{2L}{\lambda^2\|\nabla f(w_0)\|}\Big\rceil - 2\Big\}, \gamma = \frac{L}{2\lambda^2}\|\nabla f(w_0)\| - \frac{t_0}{4} \in \Big[0, \frac{1}{2}\Big);$$

then we have:

$$T_1 \leq \begin{cases}\frac{\lambda}{L}(t_0 - t + \frac{2\gamma}{1-\gamma}), & t \leq t_0 \\ \frac{2\lambda\gamma^{2^{t-t_0}}}{L(1-\gamma^{2^{t-t_0}})}, & t > t_0\end{cases} \quad (23)$$

We next bound T_2, which is due to the δ-approximation error.

$$\begin{aligned}T_2 = \|d_{t-1}^R - \hat{d}_{t-1}\| &= \|\hat{d}_{t-1} - d_{t-1,R} + d_{t-1,R} - d_{t-1}^R\| \\ &\leq \|\hat{d}_{t-1} - d_{t-1,R}\| + \|d_{t-1,R} - d_{t-1}^R\|,\end{aligned}$$

where $d_{t-1,R}$ is true Newton direction d_{t-1} obtained by using Richardson iteration after R iteration. From the Theorem 1, by considering $\hat{d}_{t-1} = x^*$, $d_{t-1,R} = x_k$, $d^R_{t-1} = \frac{1}{n}\sum_{i=1}^n x_{i,k}$, and choosing $\alpha \le \min\{\frac{1}{R}, \frac{1}{\hat{\lambda}_{max}}\}$, we have:

$$\|d_{t-1,R} - d^R_{t-1}\| \le E_2 = O\bigg(\frac{\nu}{R^2}\left(\|g_{t-1}\| + \|d_0\|\right)\bigg),$$

and

$$\|\hat{d}_{t-1} - d_{t-1,R}\| \le \|(I-\alpha A)^R\|\|\hat{d}_{t-1} + d_0\|,$$

By choosing $\|d_0| = 0$, we have:

$$\begin{aligned}
T_2 &\le \|(I-\alpha A)^R\|\|\hat{d}_{t-1}\| + O\bigg(\frac{\nu}{R^2}\left(\|g_{t-1}\|\right)\bigg)\\
&\le \|(I-\alpha A)^R\|\|\hat{d}_{t-1}\| + O\bigg(\frac{\nu}{R^2}\|H_{t-1}H^{-1}_{t-1}g_{t-1}\|\bigg)\\
&\le \|(I-\alpha A)^R\|\|\hat{d}_{t-1}\| + O\bigg(\frac{\nu}{R^2}\|H_{t-1}\|\|H^{-1}_{t}g_{t-1}\|\bigg)\\
&\le \|(I-\alpha A)^R\|\|H^{-1}_{t-1}g_{t-1}\| + O\bigg(\frac{\nu L}{R^2}\|H^{-1}_{t-1}g_{t-1}\|\bigg)\\
&\le \bigg(\|(I-\alpha A)^R\| + O\Big(\frac{\nu L}{R^2}\Big)\bigg)\|H^{-1}_{t-1}g_{t-1}\|\\
&\le \delta\|H^{-1}_{t-1}\|\|g_{t-1}\| \le \frac{\delta}{\lambda}\|\nabla f(w_{t-1})\|,
\end{aligned}$$

so

$$\eta_{t-1}T_2 \le \eta_{t-1}\frac{\delta}{\lambda}\|\nabla f(w_{t-1})\|,$$

where

$$\delta = \|(I-\alpha A)^R\| + O\Big(\frac{\nu L}{R^2}\Big),$$

In damped phase, when $t \le t_0$, $\eta_{t-1} = \frac{\lambda^2}{L\|\nabla f(w_{t-1})}\|$. In pure Newton phase, $t > t_0$ and $\eta_{t-1} = 1$. So

$$\begin{aligned}
\eta_{t-1}T_2 &\le \begin{cases} \frac{\delta\lambda}{L}, & t \le t_0\\ \frac{\delta}{\lambda}\|\nabla f(w_{t-1})\|, & t > t_0\end{cases}\\
&\le \begin{cases} \frac{\delta\lambda}{L}, & t \le t_0\\ \delta\kappa\|w_{t-1} - w^*\|, & t > t_0\end{cases}
\end{aligned} \tag{24}$$

Combine (23) and (24) we have:

$$\|w_t - w^*\| \le \begin{cases} \frac{\lambda}{L}(t_0 - t + \frac{2\gamma}{1-\gamma}) + \frac{\delta}{\kappa}, t \le t_0\\ \frac{2\lambda\gamma^{2^{t-t_0}}}{L(1-\gamma^{2^{t-t_0}})} + \delta\kappa\|w_{t-1} - w^*\|, t > t_0\end{cases}, \tag{25}$$

Appling (25) recursively we have:

$$\|w_t - w^*\| \le \begin{cases} \frac{1}{\kappa}(t_0 - t + \frac{2\gamma}{1-\gamma}) + \frac{\delta}{\kappa}, t \le t_0\\ \frac{2t\gamma^{2^{t-t_0}}}{\kappa(1-\gamma^{2^{t-t_0}})} + (\delta\kappa)^t\|w_0 - w^*\|, t > t_0\end{cases},$$

6.3 Proof of Theorem 2

From Lemma 1, if $\|w_0 - w^*\| \ge \frac{2t\gamma^{2^{t-t_0}}}{\kappa(\delta\kappa)^t(1-\gamma^{2^{t-t_0}})}$, we have:

$$\|w_t - w^*\| \le 2(\delta\kappa)^t\|w_0 - w^*\| \le \epsilon,$$

then we can obtain

$$T \ge \delta\kappa\frac{\log 2\|w_0 - w^*\|/\epsilon}{\log 1/\gamma}.$$

6.4 Proof of Theorem 3

From Lemma 1, if $\|w_0 - w^*\| < \frac{2t\gamma^{2^{t-t_0}}}{\kappa(\delta\kappa)^t(1-\gamma^{2^{t-t_0}})}$, we have:

$$\|w_t - w^*\| \le \frac{4t\gamma^{2^{t-t_0}}}{\kappa(1-\gamma^{2^{t-t_0}})} \le \epsilon,$$

then we have:

$$\begin{aligned}
\frac{\kappa(1-\gamma^{2^{t-t_0}})}{4t\gamma^{2^{t-t_0}}} &\ge \frac{1}{\epsilon}\\
\log\frac{\kappa}{4t} + \log(1-\gamma^{2^{t-t_0}}) + 2^{t-t_0}\log(\gamma) &\ge \log\frac{1}{\epsilon}
\end{aligned}$$

As $\gamma \in [0, \frac{1}{2}]$, when t increases, $\log(1-\gamma^{2^{t-t_0}}) \to 0$. Besides, $\log\frac{\kappa}{4t} << 2^{t-t_0}\log(\gamma)$. So, we have:

$$T = O\Big(\log\log\frac{1}{\epsilon}\Big).$$

References

[1] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the Edge," *Proc. IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

[2] Z. Zhou *et al.*, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[3] X. Wang *et al.*, "Convergence of Edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, Second Quarter 2020.

[4] J. Konecny *et al.*, "Federated Learning: Strategies for improving communication efficiency," in *Proc. Int. Conf. Neural Inf. Process. Syst. Workshop Private Multi-Party Mach. Learn.*, 2017.

[5] H. B. McMahan, E. Moore, D. Ramage, and S. Hampson, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

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

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

[8] Y. Tu, Y. Ruan, S. Wagle, C. G. Brinton, and C. Joe-Wong, "Network-aware optimization of distributed learning for fog computing," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2509–2518.

[9] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018.

[10] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

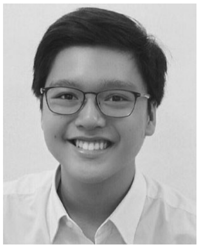
[11] S. P. Boyd and L. Vandenberghe, *Convex Optim.*, Cambridge, U.K.: Cambridge Univ. Press, 2004.

[12] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Berlin, Germany: Springer, 2013.

[13] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate Newton-Type method," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1000–1008.
[14] C. T. Dinh *et al.*, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021.
[15] S. Wang, F. Roosta, P. Xu, and M. W. Mahoney, "GIANT: Globally improved approximate newton method for distributed optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2338–2348.
[16] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Proc. Adv. Neural Inf. Process.g Syst.*, 2010.
[17] O. Shamir and N. Srebro, "Distributed stochastic optimization and learning," in *Proc. 52nd Annu. Allerton Conf. Commun. Control Comput.*, 2014, pp. 850–857.
[18] J. D. Lee, Q. Lin, T. Ma, and T. Yang, "Distributed stochastic variance reduced gradient methods by sampling extra data with Replacement," *J. Mach. Learn. Res.*, vol. 18, no. 122, pp. 1–43, 2017.
[19] S. J. Reddi, J. Konecny, P. Richtarik, B. Poczos, and A. Smola, "AIDE: Fast and communication efficient distributed optimization," 2016, *arXiv:1608.06879*.
[20] P. Richtárik and M. Takáč, "Distributed coordinate descent method for learning with big data," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2657–2681, 2016.
[21] T. Yang, "Trading computation for communication: Distributed stochastic dual coordinate ascent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 629–637.
[22] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for Wireless federated learning," 2019, *arXiv: 1911.00856*.
[23] H. H. Yang, Z. Liu, T. Q. S. Quek, and H. V. Poor, "Scheduling policies for federated learning in Wireless networks," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 317–333, Jan. 2020.
[24] H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu, "DoubleSqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6155–6165.
[25] Y. Zhang and L. Xiao, "DiSCO: Distributed optimization for self-concordant empirical loss," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 362–370.
[26] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.
[27] W. C. Rheinboldt, "Classical iterative methods for linear systems," Tech. Univ. Munich, 2009.
[28] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Berlin, Germany: Springer, 2009.
[29] B. Polyak and A. Tremba, "New versions of Newton method: Step-size choice, convergence domain and under-Determined equations," *Optim. Methods Softw.*, vol. 35, no. 6, pp. 1272–1303, Nov. 2020.
[30] M. Li *et al.*, "Scaling distributed machine learning with the parameter server," in *Proc. 11th USENIX Conf. Oper. Syst. Des. Implementation*, 2014, pp. 583–598.
[31] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 19–27.
[32] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. 7th Int. Conf. Learn. Representations*, 2018.
[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[34] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 2921–2926.
[35] T. Li *et al.*, "Federated optimization in heterogeneous networks," in *Proc. 3rd MLSys Conf.*, 2020, pp. 429–450.
[36] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proc. Eur. Symp. Artif. Neural Netw. Comput. Intell. Mach. Learn.*, 2013.

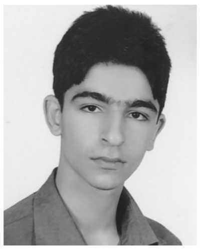
Canh T. Dinh received the BE degree in electronics and telecommunication from the Ha Noi University of Science and Technology, Ha Noi City, Vietnam, in 2015, and the master's degree in data science from Université Grenoble Alpes, Grenoble, France, in 2019. He is currently working toward the PhD degree in computer science with the University of Sydney, Sydney, Australia. His supervisor is Dr. Nguyen H. Tran. His research interests include federated learning and privacy machine learning.

Nguyen H. Tran (Senior Member, IEEE) received the BS degree in electrical and computer engineering from the HCMC University of Technology, in 2005, and the PhD degree in electrical and computer engineering from the Kyung Hee University, in 2011. He was an assistant professor with the Department of Computer Science and Engineering, Kyung Hee University, from 2012 to 2017. Since 2018, he has been with the School of Computer Science, The University of Sydney, where he is currently a senior lecturer. His research interests include distributed computing, machine learning, and networking. He received the best KHU thesis Award in engineering in 2011 and several best paper awards, including IEEE ICC 2016 and ACM MSWiM 2019. He receives the Korea NRF Funding for Basic Science and Research 2016–2023 and ARC Discovery Project 2020–2023. He was the editor of the *IEEE Transactions on Green Communications and Networking* from 2016 to 2020, and the associate editor of *IEEE Journal of Selected Areas in Communications* 2020 in the area of distributed machine learning/federated learning.



Tuan Dung Nguyen received the BS degree in computer science from the University of Melbourne, Australia. He is currently working toward the MPhil degree with the Computational Media Lab, Australian National University. His research interests include distributed optimization, machine learning and computational social science.

Wei Bao (Member, IEEE) received the BE degree in communications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009, the MASc degree in electrical and computer engineering from the University of British Columbia, Vancouver, Canada, in 2011, and the PhD degree in electrical and computer engineering from the University of Toronto, Toronto, Canada, in 2016. He is currently a senior lecturer with the School of Computer Science, University of Sydney, Sydney, Australia. His research interests include network science, with particular emphasis on Internet of things, mobile computing, and edge computing. He received the best paper awards in ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM) in 2013 and 2019 and IEEE International Symposium on Network Computing and Applications (NCA) in 2016.



Amir Rezaei Balef received the BE degree in electrical engineering from the Amirkabir University of Technology, Tehran, Iran, in 2017, and the MSc degree in field of digital systems from the Sharif University of Technology, Tehran, Iran, in 2019. His research interests include edge computing, Internet of Things, privacy-preserving machine learning, federated learning. In recent years, he has focused on optimization algorithms.

Bing B. Zhou (Member, IEEE) received the graduate degree in electronic engineering from the Nanjing Institute of Technology in China, in 1982, and the PhD degree in computer science from Australian National University, Australia, in 1989. He is currently an associate professor with the School of Computer Science, University of Sydney, Australia, in 2003. Currently, he is the theme leader for distributed computing applications in the Centre for Distributed and High-Performance Computing at the University of Sydney.

Albert Y. Zomaya (Fellow, IEEE) is currently a chair professor of High-Performance Computing and Networking with the School of Computer Science and director of the Centre for Distributed and High-Performance Computing with the University of Sydney. To date, he has published more than 600 scientific papers and articles and is (co-)author/editor of more than 30 books. A sought-after speaker, he has delivered more than 190 keynote addresses, invited seminars, and media briefings. His research interests include span several areas in parallel and distributed computing and complex systems. He is currently the editor-in-chief of the ACM Computing Surveys and served in the past as editor-in-chief of the *IEEE Transactions on Computers* (2010–2014) and the *IEEE Transactions on Sustainable Computing* (2016–2020). He is a decorated scholar with numerous accolades including Fellowship of the IEEE, the American Association for the Advancement of Science, and the Institution of Engineering and Technology, U.K. Also, he is an Elected fellow of the Royal Society of New South Wales and an Elected Foreign member of Academia Europaea. He is the recipient of 1997 Edgeworth David Medal from the Royal Society of New South Wales for outstanding contributions to Australian Science, the IEEE Technical Committee on Parallel Processing Outstanding Service Award (2011), IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing (2011), IEEE Computer Society Technical Achievement Award (2014), ACM MSWIM Reginald A. Fessenden Award (2017), and the New South Wales Premier's Prize of Excellence in Engineering and Information and Communications Technology (2019).