Resource-constrained Federated Edge Learning with Heterogeneous Data: Formulation and Analysis

Yi Liu†, *Student Member, IEEE*, Yuanshao Zhu†, *Graduate Student Member, IEEE*, and James J.Q. Yu, *Senior Member, IEEE*

Abstract—Efficient collaboration between collaborative machine learning and wireless communication technology, forming a Federated Edge Learning (FEEL), has spawned a series of next-generation intelligent applications. However, due to the openness of network connections, the FEEL framework generally involves hundreds of remote devices (or clients), resulting in expensive communication costs, which is not friendly to resource-constrained FEEL. To address this issue, we propose a distributed approximate Newton-type algorithm with fast convergence speed to alleviate the problem of FEEL resource (in terms of communication resources) constraints. Specifically, the proposed algorithm is improved based on distributed L-BFGS algorithm and allows each client to approximate the high-cost Hessian matrix by computing the low-cost Fisher matrix in a distributed manner to find a "better" descent direction, thereby speeding up convergence. Second, we prove that the proposed algorithm has linear convergence in strongly convex and non-convex cases and analyze its computational and communication complexity. Similarly, due to the heterogeneity of the connected remote devices, FEEL faces the challenge of heterogeneous data and non-IID (Independent and Identically Distributed) data. To this end, we design a simple but elegant training scheme, namely FedOVA (Federated One-vs-All), to solve the heterogeneous statistical challenge brought by heterogeneous data. In this way, FedOVA first decomposes a multi-class classification problem into more straightforward binary classification problems and then combines their respective outputs using ensemble learning. In particular, the scheme can be well integrated with our communication efficient algorithm to serve FEEL. Numerical results verify the effectiveness and superiority of the proposed algorithm.

Index Terms—Resource-constrained, Federated Edge Learning, Non-IID Data, Newton-type Methods, One-vs-All Methods.

I. INTRODUCTION

WITH the convergence of machine learning technology and wireless communication technology, various smart mobile applications have been developed to provide people with high-quality services, which have revolutionized our daily lives and brought benefits to the human societies and national economy [2]. The key idea of developing machine learning-based smart mobile applications in wireless networks is to learn (train) models leveraging the rich data collected by massively distributed user devices [3]. For example, traditional cloud computing-based centralized machine learning frameworks focus on central data processing, which generally requires widely distributed mobile devices to offload their local training data to remote cloud centers for centralized learning [4], [5]. Nevertheless, such a solution still faces two severe challenges: limited communication resources and data privacy. It is impractical for edge devices with limited resources (in terms of computation and communication resources) to offload massive amounts of local training data to the remote cloud center. On the other hand, since the remote cloud center is inevitably attacked by some malicious opponents, the private information involved in the training data uploaded by the edge device may be leaked [6]. Therefore, such a framework is no longer suitable for a modern society that values the right to privacy, motivating us to develop a new learning framework to solve the above challenges.

In light of the two challenging issues, the researchers designed a novel architecture called Mobile Edge Computing (MEC) [7], which significantly reduces communication latency and mobile energy consumption by migrating cloud computation capability and learning tasks from remote cloud center to edge servers [8], [9]. Unfortunately, previous work cannot be used to mitigate our challenging issues. The main reason is that the server in the MEC architecture still needs to access the original data of the remote device, which still leads to privacy leaks [10]. To address the privacy concern, a collaborative model training scheme was proposed, i.e., Federated Learning [11], which leverages large amounts of data distributed over multiple edge devices, such that the latter do not share locally private data with any centralized entity. Concretely, edge devices (or *clients*) are asked to upload model updates to a centralized server iteratively, thereby jointly training a shared global model. In this context, the central server can use the rich distributed data to train the optimal shared model and can rationally allocate the computation resources of the edge devices [4]. Inspired by the above facts, effective collaboration between MEC and FL (known as Federated Edge Learning (FEEL)) has great potential to promote collaborative machine

learning in next-generation smart applications.

Although the FEEL framework has achieved great success in edge computing networks and spawned a series of emerging applications such as Google Keyboard [12], it still suffers from heterogeneous data, and resource constraints (in terms of communication cost) [13]. Unlike the MEC architecture, the data at the edge is generally heterogeneous and non-independent, and identically distributed (non-IID) in the FEEL framework, which impairs the performance of the machine learning model [14]–[16]. Obviously, due to heterogeneous data challenges and non-IID data issues, resulting in slower convergence speed (i.e., more training rounds), expensive communication overhead is still the bottleneck of FEEL. Therefore, it is necessary to build a bridge to link training schemes that relieve heterogeneous data with efficient communication algorithms.

Building such a bridge inevitably needs to answer the following question: *how to develop a distributed training algorithm which uses a minimal number of communication rounds to achieve model performance comparable to that of the benchmark FL in [11] for convergence and handles non-IID and heterogeneous data across edge devices?* To answer the above question, our goal is to develop a distributed optimization algorithm to achieve faster convergence while being robust to non-IID data. Towards the goal, we present an (approximate) Newton-type method-based distributed second-order optimization algorithm. Our starting point is leveraging gradient and curvature information to find a "better" descent direction, thereby significantly speeding up the convergence speed. Standing on top of it, to address non-IID issues, we propose a simple but elegant training scheme, namely FedOVA (Federated One-vs-All), which can work effectively with the non-IID data in FEEL by decomposing the multi-classification task into multiple binary classification tasks. Specifically, our insight is to decompose a federated multiclass classification task on non-IID data into multiple binary classification tasks on the client-side by introducing the One-vs-All [17] training scheme. In particular, the main contributions of this paper are summarized as follows:

- *A communication-efficient distributed approximate Newton-type method for resource-constrained federated edge learning:* We propose a communication-efficient (in terms of convergence speed) L-BFGS optimization algorithm based on (approximate) second-order information with stochastic batches for the federated edge learning framework, as a novel approach to the empirical risk minimization problems. The classic L-BFGS with stochastic batches is generally unstable, which is not friendly for us to implement this algorithm in a distributed manner. For this reason, we use smooth estimation to evaluate the gradient difference while utilizing the Fisher Information Matrix (FIM) to approximate the Hessian matrix to accelerate the convergence.
- *A simple but elegant training scheme with heterogeneous data for federated edge learning:* We design a simple but elegant training scheme, namely FedOVA, to address the heterogeneous data and non-IID problems in FEEL. Different from existing solutions, FedOVA

addresses this problem from the perspective of learning. Specifically, FedOVA (1) improves performance and convergence speed (2) without introducing additional overhead or operations to the FEEL framework.

- *Theoretical properties and convergence analysis:* We theoretically comprehensively analyze the convergence of the proposed second-order optimization algorithm and demonstrate the effective cooperation between the Fisher Information Matrix and L-BFGS. Notably, we show that the proposed algorithm can linearly converge to the neighborhood of optimal solutions for convex and non-convex problems under standard assumptions for general empirical risk minimization problems. We replace the Hessian matrix used to approximate the gradient difference with the Fisher Information Matrix in a distributed environment. Such a method is comparable to mainstream first-order optimization algorithms.
- *Extensive experimentation and model validation:* Extensive case studies conducted on several public datasets (i.e., F-MNIST, CIFAR-10, and KWS dataset) in computer vision and natural language processing tasks have empirically demonstrated the effectiveness of the proposed method. Experimental results verify that our solution is significantly better than mainstream methods on heterogeneous and non-IID data in terms of classification accuracy and convergence time. In particular, we emphasize that our method is empirically applicable to both convex and non-convex machine learning models.

Remark: Please note that this work is an extension of the authors' prior conference publication [1]. Based on [1], we introduce notable algorithmic and theoretical improvements in this manuscript. Apart from textual amendments and changes, we thoroughly revamp and improve the system design for communication efficient and heterogeneous data in FEEL. Furthermore, we introduce FEEL's FIM-based distributed second-order optimization algorithm to reduce the communication and run-time cost significantly. Lastly, we prove that the proposed algorithm has linear convergence in strongly convex and non-convex cases and analyze its computational and communication complexity. Additional case studies are incorporated to validate the improvements.

The rest of this paper is organized as follows. We first review the related work in Sec. II. Then, we present some background knowledge of this paper in Sec. III. Sec. IV elaborates on the proposed algorithm. Sec. V explains the complexity analysis of the proposed algorithm. We conduct a series of experimental and analyses in Sec. VI. Finally, we conclude this paper in Sec. VII.

II. RELATED WORK

A. Federated Edge Learning

Federated Edge Learning (FEEL) is an emerging mobile edge computing framework that integrates the advantages of federated learning (artificial intelligence) and mobile edge computing, which arouses the research interest of researchers. The survey [15] comprehensively introduced FEEL's architecture, the state-of-the-art techniques, standards, and case

studies and gave some future research directions. Although there are many unfilled research gaps in the FEEL field, the academic and industrial circles generally focus on topics such as resource optimization [5], [18], communication efficiency optimization [19], [20], statistical heterogeneous optimization [21], [22], and security and privacy [6], [23]. In terms of resource optimization, researchers generally utilized economic tools (e.g., contract theory [24], auction theory [2]), and designed some incentive mechanisms [25] to implement resource management [18], resource allocation [5], and resource scheduling [26] in FEEL. For communication efficiency optimization, they utilized compression [27] and quantization [28] techniques to reduce the communications volume, thereby improving the communication efficiency of the FEEL framework. Furthermore, the FEEL framework's security and privacy research topics have never left the researchers' sight. As for the present, some common malicious attacks against the FEEL framework such as poisoning attacks [29], [30], backdoor attacks [31], [32], adversarial attacks [33], and gradient leakage attacks [34] have been extensively studied. In the future, more new problems and new challenges will be raised and solved by researchers.

B. Distributed Optimization Methods in Federated Edge Learning

1) First-order distributed optimization methods: The first-order distributed optimization method is the most common and practical in a distributed learning environment, in which it mainly leverages gradient information to find a descent direction to update the global model iteratively. Distributed Stochastic Gradient Descent (D-SGD) [35] as a classic example of this type of algorithm has derived a series of variants such as ColumnSGD [36], variance-reduced SGD [37] and so on. However, these algorithms sacrifice communication rounds in exchange for accuracy, which is unfriendly to the resource-constrained FEEL framework. And it implies that communication overhead is still a bottleneck problem in the resource-constrained FEEL framework.

2) Second-order distributed optimization methods: Unlike the classic first-order distributed optimization method, the second-order distributed optimization method utilizes gradient information and second-order information (i.e., curvature information) to find a "better" descent direction to update the global model iteratively. However, previous work [38] pointed out that the complex inverse matrix-vector product involved in the calculation of curvature information is not suitable for distributed environments. To this end, in [13], [39], [40], the authors designed DANE, DONE, and GIANT, respectively, to overcome this problem. Specifically, DANE circumvented the challenges mentioned above by designing a well-designed local optimization problem, while DONE proposed a distributed approximate Newton method based on the Richardson iteration. GIANT used the harmonic mean Hessian matrix to approximate the true Hessian matrix, significantly reducing communication costs. Similar to GIANT, we focus on how to approximate the Hessian matrix to reduce communication costs efficiently. This paper develops a variant of the L-

BFGS algorithm that uses the Fisher information matrix to approximate the Hessian matrix to achieve fast convergence.

C. Non-IID Issues in Federated Edge Learning

Statistical challenges (i.e., heterogeneous data and non-IID data) are unresolved issues in the FEEL framework. The gradient-based aggregation rules (e.g., FedAvg [11]) rely on D-SGD, which is widely used to iteratively train deep learning models under the assumption of IID training data. The purpose of learning from IID training data is to ensure that the stochastic gradient is an unbiased estimate of the full gradient [22]. In practice, however, it is unrealistic to assume that the local data on each edge client are always IID.

Existing works have addressed the non-IID issue in FEEL by designing additional framework mechanisms. For example, Zhao *et al.* in [22], [35] introduced a data-sharing mechanism to improve FedAvg in non-IID data settings. The proposed mechanism involves distributing a small amount of globally shared data containing examples of each class, thereby introducing a trade-off between accuracy and centralization. However, this approach unintentionally discloses the client's private data as the dataset is publicly shared, hence violating FL's privacy protection requirement. Another popular solution is to design a performance-oriented client selection mechanism. Kopparapu *et al.* in [41] proposed FedCD, an aggregation method that can clone and delete models to dynamically group devices with similar data, thereby selecting the clients with high-quality updates to mitigate the non-IID issue. However, the server needs to calculate the model quality score in each round to decide whether to clone or delete it, introducing additional computational overhead. Nonetheless, in real-world applications, applying reinforcement learning within resource-constrained FL environments is likely to be impractical. However, how to organically combine communication efficient algorithms with non-IID robust algorithms is still an open question.

III. PRELIMINARIES

A. Federated Edge Learning Pipeline

In the FEEL setting, we consider a server $\mathcal{S}$ and K clients, participating in training a shared global model ω^* without sharing their raw private data. In this context, we assume that each client holds a local training dataset $\mathcal{D}_k = \{x_i, y_i\}_{i=1}^{n_k}$, where n_k denotes the number of samples. For the model parameters $\omega \in \mathbb{R}^d$ and a local training dataset $\mathcal{D}_k$, let $F_k(\omega, x_i)$ be the loss function at the client, and let $f(\omega)$ be the loss function at the server. Accordingly, the pipeline of federated learning is defined as follows.

Phase 1, initialization: First, the server selects a certain proportion q of clients from all clients to participate in an FL learning task. Second, the server broadcasts the initialized global model ω_0 to all clients, i.e., $\omega_0^k \leftarrow \omega_0$.

Phase 2, local training: For the t-th training round, each client trains the received global model ω_t^k on its own local dataset $\mathcal{D}_k$. On the client side, the goal is to minimize the following objective function: $F_k(\omega) = \frac{1}{n_k} \sum_{i \in \mathcal{D}_k} \ell_i(\omega)$, where

$\mathcal{D}_k$ is the set of indexes of data samples on the client k. For a given federated learning task (such as image classification task), we typically take cross-entropy loss:

$$f_i(\omega) = \ell(x_i, y_i; \omega) = -\sum_{m=1}^{M} \left(I_{[y_i=m]} \log \sigma_m(x_i; \omega) \right), \quad (1)$$

where $\sigma(x_i; \omega)$ is the Softmax operator and m is the number of classe, as the loss function of the local client, i.e., the loss of the prediction on training example (x_i, y_i) made with model parameters ω. Furthermore, the gradient and Hessian matrix of $F_k(\omega)$ can be calculated as:

$$\nabla F_k(\omega) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\omega), H_k(\omega) = \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(\omega). \quad (2)$$

Then each client uploads its model updates $\Delta \omega_k = \eta \nabla F_k(\omega)$, where η is the learning rate, to the server.

Phase 3, aggregation: The server uses a model updates aggregation rule like FedAvg [11] to aggregate all the updates to obtain a new global model ω_{t+1}. Specifically, on the server side, the goal is to minimize the following objective function:

$$f(\omega) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(\omega), F_k(\omega) = \frac{1}{n_k} \sum_{i \in \mathcal{D}_k} \ell_i(\omega). \quad (3)$$

Note that the above steps will be terminated until the global model reaches convergence.

B. Challenges to Distributed Newton's Method

In this paper, our goal is to utilize the Newton's method [13], [39] to efficiently solve the problem of minimizing the empirical risk function in the FEEL framework, thus, we run:

$$\begin{aligned}
\omega_{t+1} &= \omega_t - \left(\nabla^2 f(\omega_t) \right)^{-1} \nabla f(\omega_t) \\
&= \omega_t - \left(\frac{1}{K} \sum_{i=1}^{K} \nabla^2 F_k(\omega_t) \right)^{-1} \left(\frac{1}{K} \sum_{i=1}^{K} \nabla F_k(\omega_t) \right),
\end{aligned} \quad (4)$$

where $\nabla^2 F_k(\omega_t)$ and $\nabla F_k(\omega_t)$ are respectively the Hessian information $\mathbf{H}_t \in \mathbb{R}^{d \times d}$ and gradient information computed by each client. To be specific, the server collects and aggregates $\nabla^2 F_k(\omega_t)$ and $\nabla F_k(\omega_t)$ uploaded by these clients to update the global model (i.e., update Eq. (3)). *However, there is a big drawback: the calculation of the Hessian is computationally expensive.* Hence, the core requirement of designing the distributed Newton's method is *Hessian-free communication* and *inverse-Hessian-free computation.* We assume that the size and feature vector dimensions of the client's local dataset are $\mathcal{D}_k$ and d, respectively, so that each client needs to send Hessians with size $\mathcal{O}(d^2)$ over the network or compute the inverse Hessian with complexity $\mathcal{O}(\sum_{i=1}^{n} \mathcal{D}_i d^2 + d^3)$. This is obviously unrealistic for the resource-constrained FEEL framework. In this paper, to address this challenge, we design an FIM-based distributed L-BFGS method for the resource-constrained FEEL framework to approximate Hessian at a low cost and achieve fast convergence.
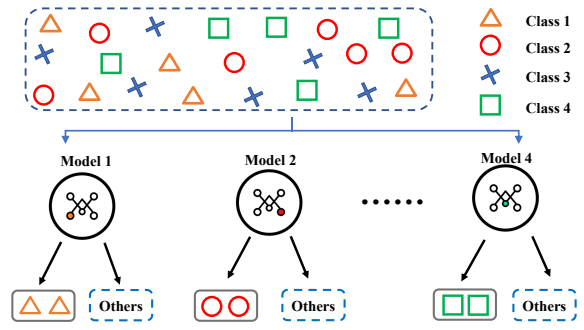

Fig. 1. The OVA training scheme decomposes an n-class ($n > 2$) classification task into n binary classification tasks, in this figure, $n = 4$.

C. One-vs-All Training Scheme

One-vs-All (OVA) [42] training scheme is generally used in logistic regression to solve multi-classification problems. For example, for an n-class ($n > 2$) classification task, we assume a labeled dataset $D = \{x_i, y_i\}_{i=1}^{n}$ where $y_i \in \{1, 2, \ldots, n\}$ is the ground-truth label of x_i and n is the total number of samples. As shown in Fig. 1, the OVA training scheme trains n binary classifiers $f_i(\cdot), i \in \{1, 2, \ldots, n\}$, with each classifier discriminating one class from the others in D. Each classifier can be expressed as follows:

$$f_i(x) = P(y = i | x; \omega_i), \quad (5)$$

where ω_i refers to the parameters of classifier $f_i(\cdot)$. To obtain a prediction for a new instance, OVA uses these n binary classifiers to calculate the confidence that the instance belongs to the current class. Specifically, we select the prediction with the highest confidence as the final classification for this new instance. This process is formalized as follows:

$$\hat{y} = \underset{i \in \{1,2,\ldots,n\}}{\arg\max} f_i(x). \quad (6)$$

Remark: It is important to note that, although OVA is a general-purpose approach to multi-class classification that has been used for decades, existing OVA-based works have focused on centralized learning. In this work, we instead leverage distributed OVA to tackle the open problem of distributed multi-class classification under non-IID data. Specifically, we take advantage of the independence of each binary classifier in OVA and incorporate it into FEEL to propose a novel training scheme named FedOVA. This scheme is able to solve the non-IID data problem in FEEL efficiently, as will be described in Section IV. To the best of our knowledge, this is the first work that integrates FEEL and OVA to address FEEL with non-IID data. Moreover, this solution will not affect the normal execution of the communication efficient algorithm, which opens a door for solving the non-IID problem and the expensive communication overhead problem at the same time.

IV. OUR ALGORITHM DESIGN

In this section, we first present our FIM-based approximate distributed optimization algorithm, and then introduce the designed FedOVA training program in detail. Second, the summaries of symbols are presented in Table I.

TABLE I
SUMMARIES OF APPLIED SYMBOLS.

Symbols	Description
$\mathcal{D}_k$	The k-th client local dataset
d	The feature dimensions of the data
$\mathcal{K}_t$	The set of clients selected at round t
ω	The global model
q	The client participation rate ($0 < q < 1$)
K	The total number of clients
k	The number of clients participating in training
$\mathbf{H}_t$	The Hessian information
Γ	The diagonal matrix of Hessian information
p	The direction of gradient descent
η	The learning rate
$\nabla f(\omega)$	The gradient information
S_t	The stochastic batch picked at iteration t
m	The memory size
B_t	The Hessian information at round t
$\mathcal{B}_t$	The Hessian approximation information
$J(\omega)$	$J(\omega)$ is the Jacobian matrix of $f(\omega)$ with respect to ω
τ	The number of blocks
B	The local min-batch size
E	The number of local epochs
T	The round of iteration
ω_t^k	The local model parameter of the k-th client at round t

A. FIM-based Approximate Distributed Optimization Algorithm

In the distributed learning community, researchers generally use the classic L-BFGS [43] optimization algorithm with stochastic batches to update Eq. (4). We first review stochastic L-BFGS algorithm. Given an optimization problem with d variables, L-BFGS only needs to store a few vectors of length d to implicitly approximate the Hessian. Therefore, the intrinsic idea within L-BFGS is to utilize the curvature information implied by the vector pairs (s_t, y_t) to help regularize the gradient direction, where $s_t = \omega_{t+1} - \omega_t$ and $y_t = \nabla f(\omega_{t+1}) - \nabla f(\omega_t)$. Specifically, the L-BFGS algorithm needs the leverage history state, i.e., the last m (generally $m = 10$) elements in (s_t, y_t) to update the current state. In the stochastic L-BFGS algorithm, y_t is replaced by a stochastic version of $B_t^{S_t}(\omega_{t+1} - \omega_t)$, i.e.,

$$y_t = B_t^{S_t}(\omega_{t+1} - \omega_t), \qquad (7)$$

$$B_t^{S_t} \stackrel{def}{=} \frac{1}{|S_t|} \sum_{i \in S_t} \nabla^2 f_i(\omega_t), \qquad (8)$$

where S_t is the stochastic batch picked at iteration t. Then we can use two-loop recursion based vector-free L-BFGS (VL-BFGS) algorithm which proposed in [44] to update the vector pairs (s_t, y_t). In this way, we can achieve the necessary goals of designing distributed second-order algorithms mentioned above, i.e., Hessian-free communication and inverse-Hessian-free computation. However, such a method is difficult to stabilize the algorithm, and the current measures to remedy this problem require the batch size to be large enough, which is obviously unrealistic in FEEL [45].

Motivated by the inconvenience mentioned above, in this paper, we propose a low-cost Hessian approximation method, i.e., Fisher information-based Hessian approximation method, to approximate the Hessian matrix efficiently. Specifically, according to the definition of Generalized Gauss-Newton matrix (GGN) [46], [47], Eq. (8) can be written as follows:

$$B_t^k \approx [J(\omega_t^k)]^\top \nabla^2 f(\omega_t^k) J(\omega_t^k) \stackrel{def}{=} \mathcal{B}_t^k, \qquad (9)$$

where $J(\omega_t^k)$ is the Jacobian matrix of $f(\omega)$ with respect to ω at ω_t^k and we use $\mathcal{B}_t^k$ to denote the Hessian or Hessian approximation used to smoothen y_t^k, i.e., $y_t^k = \mathcal{B}_k s_k$. Since we use a stochastic version of the gradient to update the model, we consider a stochastic batch of S_k, thus, Eq. (9) can be rewritten as:

$$\mathcal{B}_t^{S_k} = \frac{1}{|S_k|} \sum_{i \in S_k} [J(\omega_t^{S_k})]^\top \nabla^2 f_i(\omega_t^{S_k}) J(\omega_t^{S_k}). \qquad (10)$$

Recall that, since the cross-entropy loss we use is negative log-likelihood, it is not difficult to obtain $\mathbb{E}_{x \in \mathcal{D}_k}[\nabla f_k(\omega^*, x)\nabla f_k(\omega^*, x)^\top]$, where ω^* is the true parameter (i.e., the Softmax distribution of the local model) obtained in the form of Fisher information. According to the definition of two equivalent methods for calculating the Fisher information matrix [48], [49], the above mentioned equation can also be written as $\mathbb{E}_{x \in \mathcal{D}_k}[\nabla^2 f_k(\omega^*, x)]$. This means we can use $\nabla f_k(\omega, x)\nabla f_k(\omega, x)^\top$ as an asymptotically unbiased estimation of $\nabla^2 f_k(\omega, x)^\top$, since w gradually converges to ω^* during the training. Thus, our proposed Fisher information-based Hessian approximation method is defined as follows:

$$\mathcal{B}_t^{S_k} = \frac{1}{|S_k|} \sum_{i \in S_k} [J(\omega_t^{S_k})]^\top \nabla f_i(\omega_t^{S_k})\nabla f_i(\omega_t^{S_k})^\top J(\omega_t^{S_k}). \qquad (11)$$

Additionally, to significantly reduce the computational complexity and storage while maintaining the accuracy, in DNN we leverage the diagonalization technique [45] to approximate the Hessian matrix, i.e., $\Gamma = \mathrm{diag}(\nabla f_i(\omega_t^{S_k})\nabla f_i(\omega_t^{S_k})^\top) \approx \nabla f_i(\omega_t^{S_k})\nabla f_i(\omega_t^{S_k})^\top$. Thus, we only need to store the diagonal elements of FIM and make all other elements zero. Therefore, in the proposed FIM-based L-BFGS algorithm, each client computes the diagonal matrix of FIM locally and uploads it to the server. The server collects and aggregates these FIM-vector products to update the vector pair (s_t, y_t). In particular, the communication cost in each round can be $\mathcal{O}(d)$ (for more analysis details, see below). In this way, the proposed algorithm avoids Hessian communication and inverse Hessian computation.

We present our FIM-based Approximate L-BFGS Algorithm, as shown in Algorithm 1. As shown in lines 2 to 4, the server sends the initialized global model to the selected client. Then the server aggregates the FIM uploaded from the local client and utilizes the L-BFGS algorithm to update the pair (s_t, y_t), as shown in lines 5 to 15. In addition, all the local computation steps of the client are presented in the ClientUpdate function.

B. FedOVA Training Scheme

In this section, we present a detailed training procedure on how to combine OVA and FEEL, i.e., FedOVA (see Algorithm 2). We then explain why the proposed FedOVA can address the statistical challenges of federated optimization with non-IID data in a multi-classification task. Recall that the non-IID

Algorithm 1: FIM-based Approximate L-BFGS Algorithm

Input: $\mathcal{K}$ is the client set with indexed by k, S_t^k is the stochastic batch of client k at iteration t, integer history size $m > 0$.

Output: Optimized global model ω^*.

Server:

1: Choose H_0
2: **for** each round $t = 1, \ldots, T$ **do**
3:    $\mathcal{K}_t \leftarrow$ (Sample a subset of clients from $\mathcal{K}$)
4:    Server sends the ω_t to $\mathcal{K}_t$ //**Communication**
5:    **while** global model no converge **do**
6:       Compute a direction $p_t = -H_t \nabla f(\omega_t)$ by using two-loop recursion algorithm [44]
7:       Compute $\omega_{t+1} = \omega_t + \eta p_t$
8:       Update the curvature pairs: $s_t = \omega_{t+1} - \omega_t$,
      $y_t = (\frac{1}{K} \sum_{k=1}^{K} \mathcal{B}_t^{S_t^k})(\omega_{t+1} - \omega_t)$, computed by ClientUpdate(k, ω_t) function   //**Computation**
9:       **if** $k >= m$ **then**
10:          Discard vector pair (s_{t-m}, y_{t-m}) from memory storage
11:       **else**
12:          Store the vector pair (s_t, y_t)
13:       **end if**
14:    **end while**
15: **end for**

ClientUpdate (k, ω):   // **Run on client k**

1: **for** all edge clients $k = 1, \ldots, K$ **in parallel do**
2:    **for** each epoch $e = 1, \ldots, E$ **do**
3:       **for** batch $\mathcal{S}_t^k \in \mathcal{B}$ **do**
4:          $\mathcal{B}_t^{S_t^k} = \frac{1}{|S_k|} \sum_{i \in S_t^k} [J(\omega_t^{S_t^k})]^\top \operatorname{diag}(\Gamma) J(\omega_t^{S_k})$
5:       **end for**
6:    **end for**
7: **end for**
8: **return** $\mathcal{B}_t^{S_t^k}$ to server   // **Communication**

Algorithm 2: FedOVA training scheme

Input: Client set $\mathcal{K}$, n component models $\{\omega^i\}_{i=1}^n$
Output: n optimal binary classifiers $\{f_1, \ldots, f_n\}$

Server:

1: Initialize parameters of all component models ω_0
2: **for** each round $t = 1, \ldots, T$ **do**
3:    $\mathcal{K}_t \leftarrow$ (Sample a subset of clients from $\mathcal{K}$)
4:    Send the parameters w_t^i to $\mathcal{K}_t$
5:    **for** each client $k \in \mathcal{K}_t$ **do**
6:       ClientUpdate(k, ω_t^i) (refer to Algorithm 1)
7:    **end for**
8:    **for** each component model ω_t^i in group P_i **do**
9:       Update the model ω_{t+1}^i by using Algorithm 1
10:    **end for**
11: **end for**

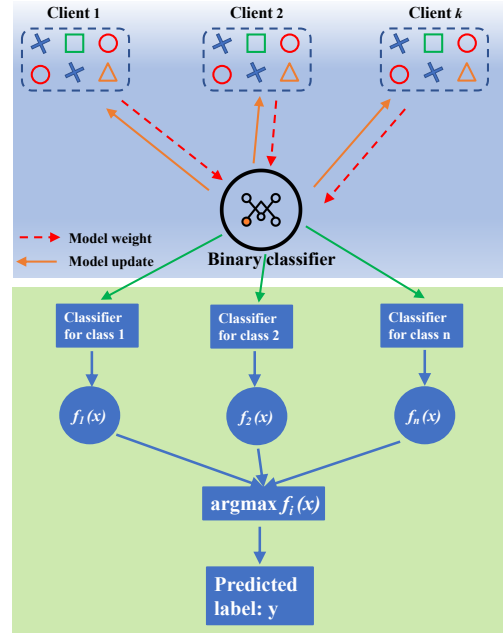problem in FEEL means that clients cannot obtain enough

Fig. 2. Overview of the FedOVA training scheme, which trains n binary classifiers and selects the output of the most confident classifier as the prediction result.

labels of multiple classes through data sharing and thus cannot train a high-precision multiclass classification model, thereby limiting the scalability of the FEEL framework. We assume that there is a server responsible for the coordination of a client set $\mathcal{K}$ in FEEL, where each client has a local dataset $\mathcal{D}_k, k \in \{1, 2, \ldots, K\}$ and the total number of classes is n. The training procedure of FedOVA repeats the following steps for communication rounds 1 to T:

Step 1, initialization: For each round of training, the server randomly selects a fraction q $(0 < q < 1)$ of the client set $\mathcal{K}$ to participate in federated edge training, resulting in subset $\mathcal{K}_t$. Then, the server broadcasts the binary classifier model parameters ω_t^i to the clients in $\mathcal{K}_t$, where $i \in \{1, 2, \ldots, n\}$ denotes the classifier ID.

Step 2, local training: After receiving the binary classifier model parameters ω_t^i, each client initializes some of the OVA component classifier models according to its own local data label distribution, i.e., $\omega_t^{i,k} \leftarrow \omega_t^i$. Taking the F-MNIST dataset as an example, if client k only has label "1" and label "2", then this client initializes the parameters of classifiers f_1 and f_2. For each binary classifier, the goal is to minimize the following objective function:

$$\arg \min_{\omega} \mathcal{L}(\omega) = \frac{1}{|\mathcal{D}_k|} \sum_{\{x_i, y_i\} \in \mathcal{D}_k} \ell(y_i, f_i(x_i; \omega)), \quad (12)$$

Each client performs stochastic gradient descent to optimize its classifiers by using the local dataset and then sends the parameter updates back to the server.

Step 3, aggregation: Since each client trains only some of the classifiers, thereby not returning parameters for all of them, and given that all classifiers are independent of one another, we can perform asynchronous updates to reduce

the computational burden. The server groups the returned parameters according to their corresponding binary classifier models and then aggregates the parameters of each group P_i. For model f_i, the aggregation process can be formulated as:

$$\omega_{t+1}^i = \frac{1}{|P_t^i|} \sum_{k \in P_t^i} \omega_t^{i,k}. \tag{13}$$

Note that the above steps are repeated until the final ensemble classifier achieves convergence.

Remark: FedOVA aims to train an "expert" binary classifier for each class, hence alleviating the problem of multi-classifier model convergence with non-IID data. Combined with FL, conventional OVA can be used to solve non-IID data problems due to the following advantages of FedOVA:

- **Training is independent.** FedOVA training is independent in the following two aspects: (1) Unlike multi-class classifier ensembles, each classifier in FedOVA is independent and specialized in distinguishing a specific class. Such a design can ensure low error correlation among different classifiers, thus enhancing their collective diversity and improving the overall classification accuracy [50]. (2) The execution of FedOVA does not affect the execution of Algorithm 1, so it can be organically integrated with Algorithm 1.
- **Comparable accuracy.** As an ensemble learning scheme, FedOVA is capable of achieving high classification accuracy. [42] has shown that when its binary classifiers are well tuned, OVA can achieve accuracy on par with any other multi-classification approach.
- **No requirement for all sample classes.** Since each binary classifier in FedOVA classifies samples as either belonging to the current class or not, missing negative sample classes does not significantly affect classification performance. When new classes emerge, FedOVA just needs to create a new classifier for each, thereby allowing for smooth adaptation to environmental changes during federated training.

V. CONVERGENCE AND COMPLEXITY ANALYSIS

In this section, we conduct a comprehensive analysis of the convergence and complexity of the proposed algorithm. First, we provide convergence theoretical analysis for both strongly convex and non-convex cases. Second, we focus on analyzing the local computing and system communication complexity of the proposed algorithm.

A. Convergence Analysis

In our settings, we use the stochastic batches of the FIM-based L-BFGS algorithm with a fixed learning rate, so the algorithm can only converge to the neighborhood of optimal point. Thus, we assume that the function $f : \mathbb{R}^d \to \mathbb{R}$ is Λ-Lipschitz continuous or Λ-smooth, i.e., $\forall i$, we have:

$$||\nabla f(\omega') - \nabla f(\omega)|| \leq \Lambda ||\omega' - \omega||, \forall \omega, \omega' \in \mathbb{R}^d, \tag{14}$$

or equivalently,

$$f(\omega') \leq f(\omega) + \nabla f(\omega)^T (\omega' - \omega) + \frac{\Lambda}{2} ||\omega' - \omega||, \forall \omega, \omega \in \mathbb{R}^d. \tag{15}$$

1) Strongly Convex Case: To analyze the convergence of the strong convex case, we need to make the following assumptions hold:

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.,*

$$\lambda I \leq \mathcal{B}^S \leq LI, \tag{16}$$

where $I \in \mathbb{R}^d$ and $\mathcal{B}$ is the Hessian (approximation) to stabilize y_t. In particular, for all batches S of size b, the above equation holds.

Assumption 2. *For all batches S and $\omega \in \mathbb{R}^d$, $\nabla f^S(\omega)$ is an unbiased estimator of the true gradient $\nabla F(\omega)$, i.e.,*

$$\mathbb{E}[\nabla f^S(\omega)] = \nabla f(\omega). \tag{17}$$

According to Assumption 1 and Eq. (15), we have:

$$f(\omega') \leq f(\omega) + \nabla f(\omega)^\top (\omega' - \omega) + \frac{\lambda}{2} ||\omega' - \omega||, \forall \omega, \omega \in \mathbb{R}^d. \tag{18}$$

Based on the above assumptions, we can state the following lemma that the Hessian approximation formulated by Algorithm 1 are bounded above and away from zero.

Lemma 1. *If Assumptions 1–2 hold, then there are two constants θ_1 and θ_2 where $0 < \theta_1 \leq \theta_2$, so that the $\mathbf{H}_t$ generated by Algorithm 1 satisfies the following equation:*

$$\theta_1 I \leq \mathbf{H}_t \leq \theta_2 I, \forall t \in \{0, 1, \dots, T\}. \tag{19}$$

Thanks to Lemma 1, we can prove the following theorem without making bounded assumptions on stochastic gradients (i.e., $||\nabla f(\omega)|| \leq \epsilon$):

Theorem 1. *Let ω^* be the minimizer of $f(\omega)$, i.e., $f^* = f(\omega^*) = \arg\min_{\omega \in \mathbb{R}^d} f(\omega)$. We assume that ω_t is the model parameter iteratively generated by the FIM-based stochastic L-BFGS algorithm with a constant learning rate $\alpha \in (0, \frac{\lambda\theta_1}{\mu})$ where $\mu = \theta_2^2 (\lambda + \Lambda\beta(b))\Lambda$. If the assumption 2 holds and f is convex, $\forall t \in \{0, 1, \dots, T\}$, we have:*

$$\begin{aligned}
\mathbb{E}[f(\omega_t) - f^*] &\leqslant \{1 - [1 - 2\alpha(\lambda\theta_1 - \alpha\mu)\}\frac{\alpha\theta_2^2 \Lambda N}{4(\lambda\theta_1 - \alpha\mu)} \\
&\quad + [1 - 2\alpha(\lambda\theta_1 - \alpha\mu)]^t [f(\omega_0) - f^*] \\
&\leqslant \frac{\alpha\theta_2^2 \Lambda N}{4(\lambda\theta_1 - \alpha\mu)} (T \to \infty),
\end{aligned} \tag{20}$$

where $\beta(b) = \frac{n-b}{b(n-1)}$ and $N = 2\mathbb{E}[||\nabla f_i(\omega^*)||^2]$.

2) Non-convex Case: Similarly, to analyze the convergence of non-convex cases, we need to make the following assumptions:

Assumption 3. *The function $f : \mathbb{R}^d \to \mathbb{R}$ is twice continuously differentiable and L-smooth, i.e.,*

$$\mathcal{B}^S \leq LI, \tag{21}$$

where $I \in \mathbb{R}^d$ and $\mathcal{B}$ is the Hessian (approximation) to stabilize y_t. In particular, for all batches S of size b, the above equation holds.

Assumption 4. *For all batches S and $\omega \in \mathbb{R}^d$, the function $f(\omega)$ is bounded below by a scalar $\hat{f}$.*

Assumption 5. *For all batches S and $\omega \in \mathbb{R}^d$, different from the strong convex case, here we need to make the bounded gradient assumption that there are constants $\gamma \geq 0$ and $\eta > 0$ that make $\mathbb{E}_S[||\nabla f^S(\omega)||^2] \leq \gamma^2 + \eta||\nabla F(\omega)||^2$ hold.*

Assumption 6. *For all batches S and $\omega \in \mathbb{R}^b$, $\nabla f^S(\omega)$ is an unbiased estimator of the true gradient $\nabla f(\omega)$, i.e.,*

$$\mathbb{E}[\nabla f^S(\omega)] = \nabla f(\omega). \tag{22}$$

Similar to strongly convex case, we also can state the following lemma that the Hessian approximation formulated by Algorithm 1 are bounded above and away from zero.

Lemma 2. *If assumptions 3–6 hold, then there are two constants θ_1 and θ_2 where $0 < \theta_1 \leq \theta_2$, so that the $\mathbf{H}_t$ generated by Algorithm 1 satisfies the following equation:*

$$\theta_1 I \leq \mathbf{H}_t \leq \theta_2 I, \forall t \in \{0, 1, \ldots, T\}. \tag{23}$$

Thanks to Lemma 2, we can prove the following theorem as follows:

Theorem 2. *Let ω^* be the minimizer of $F(\omega)$, i.e., $F^* = F(\omega^*) = \arg\min\limits_{\omega \in \mathbb{R}^d} F(\omega)$. We assume that ω_t is the model parameter iteratively generated by the FIM-based stochastic L-BFGS algorithm with a constant learning rate $\alpha \in (0, \frac{\theta_1}{\theta_2^2 \eta \Lambda})$. If the assumption 3–6 hold, $\forall t \in \{1, 2, \ldots, T\}$, we have:*

$$\mathbb{E}[\frac{1}{T} \sum_{t=0}^{T-1} ||\nabla f(\omega_t)||^2] \leqslant \frac{\alpha \theta_2^2 \gamma^2 \Lambda}{\theta_1} + \frac{2[f(\omega_0) - f^*]}{\alpha \theta_1 T} \\ \leqslant \frac{\alpha \theta_2^2 \gamma^2 \Lambda}{\theta_1} (T \to \infty). \tag{24}$$

B. Complexity Analysis

In our setting, the communication costs of the proposed algorithm involves the evaluation of $\mathcal{B}_t^{S_t}$, $\nabla f^{S_t}(\omega)$, and Algorithm 1. We state that the distributed optimization and communication cost theorem in the proposed system is as follows.

Assumption 7. *Diagonal Hessian of $\mathbf{H}_t$. The Hessian of the loss function $\mathcal{L}(\omega)$ with respect to the $\nabla^2 f(\omega)$ is always diagonal.*

Since we use the stochastic version of gradient descent to update the model, we consider a stochastic batch S_k. In this context, we assume that the batch is split into τ blocks, where the blocks are represented as $S_{k_1}, S_{k_2}, \ldots, S_{k_\tau}$ and assuming that the corresponding Jacobian block matrix is $J^{S_{k_1}}, J^{S_{k_2}}, \ldots, J^{S_{k_\tau}}$. Based on Assumption 7, the product of the Hessian vector and any vector v can be written as follows:

$$\mathcal{B}_k^{S_k} v = \sum_{i=1}^{\tau} [J(\omega_k)^{S_{k_i}}]^\top \nabla^2 f(\omega_k)^{S_{k_i}} J(\omega_k) v \\ = \sum_{i=1}^{\tau} \mathcal{B}_k^{S_{k_i}} v \tag{25}$$

Theorem 3. *If Assumption 7 holds, the total communication cost of each iteration of the proposed FIM-based L-BFGS*

algorithm is $\mathcal{O}(d\log(\tau) + m^2)$, where $\tau \geqslant m^2 + m$ is the number of clients and m is the memory size.

We prove the above theorem as follows:

Proof. First, we analyze the communication costs required for evaluating $g_t = \nabla f^{S_t}(\omega_t)$. The communication cost of the server broadcasting ω_t to k clients is $\mathcal{O}(d)$ and the communication cost of obtaining the sum of local gradients from the clients is $\mathcal{O}(d\log(\tau))$. Second, the clients need to store vector pairs $\{(s_t, y_t)\}, \{(s_t, g_t)\}$ for each iteration and compute every FIM dot-products defined in Eq. (10). To evaluate $y_t = \mathcal{B}_t^{S_{t,k}} s_t = \sum_{k=1}^{K} \mathcal{B}_t^{S_{t,k}} s_t$, the server needs to broadcast the new s_t to all clients with a communication cost of $\mathcal{O}(d)$ and receive the sum of local computation results $\mathcal{B}_t^{S_{t,k}} s_t$ with a communication cost of $\mathcal{O}(d\log(\tau))$. Then the server broadcasts it again with a communication cost of $\mathcal{O}(d)$. Therefore, the system needs to spend a total of communication cost $\mathcal{O}(d + d\log(\tau) + d) = \mathcal{O}(d\log(\tau))$ to execute the whole procedure. Third, after calculating the FIM dot-products locally on the client, the client needs to upload it to the server to execute Algorithm 1, where the communication cost is $\mathcal{O}(m^2 + m) = \mathcal{O}(m^2)$. Then, the server needs to perform a linear search on the vector pair (s_t, y_t) to update y_t, where the communication cost is $\mathcal{O}(m + d)$. Therefore, the whole procedure has a total communication cost of $\mathcal{O}(m^2 + d + m) = \mathcal{O}(m^2 + d)$. Hence, the total communication cost of each iteration of the proposed FIM-based L-BFGS algorithm is: $\mathcal{O}(d\log(\tau) + d\log(\tau) + m^2 + d) = \mathcal{O}(d\log(\tau) + m^2)$. $\square$

Comparison with FedAvg-type SGD method: In FEEL, FedAvg-type GD (or SGD) is a common aggregate approach to update the global model. In this setting, the communication cost of the server broadcasting ω_t to k clients is $\mathcal{O}(d)$ and the communication cost of obtaining the sum of local gradients from the clients is $\mathcal{O}(kd)$. Therefore, the whole procedure has a total communication cost of $\mathcal{O}(kd + d) = \mathcal{O}(kd)$. For this reason, we can see that the communication complexity of the proposed algorithm is smaller than that of FedAvg-type SGD. In particular, compared to the commonly used methods in FEEL, our method has a linear acceleration, which is friendly to resource-constrained FEEL frameworks.

VI. EXPERIMENTS

To evaluate the performance of our method under non-IID data settings, we conduct extensive experiments on three representative public datasets. All experiments were developed using Python 3.7 and PyTorch 1.7 [51], and were executed on a server with an NVIDIA GeForce RTX2080 Ti GPU and an Intel Xeon Silver 4210 CPU. Note that all experiments were performed sequentially to mimic distributed training.

A. Experimental Setup

We conduct our experiments on three established datasets: F-MNIST [52], CIFAR-10 [53], and the Speech Commands dataset [54]. The first two are image datasets with 10 classes and have been widely used in FL benchmarks. The Speech Commands dataset contains 35 classes of 1-second audio samples stored in WAV format. We select 10 common keywords

TABLE II
PERFORMANCE COMPARISON WITH DIFFERENT DISTRIBUTED
OPTIMIZATION ALGORITHMS

Method	Dataset	Training Round	Accuracy (%)
FedAvg-based SGD	F-MNIST	200	98.2
	CIFAR-10	200	79.2
	KWS	250	94.4
FedAvg-based Adam	F-MNIST	200	98.1
	CIFAR-10	200	77.9
	KWS	250	92.8
FedDANE	F-MNIST	150	97.8
	CIFAR-10	125	75.9
	KWS	200	94.3
Our Method	F-MNIST	**50**	**97.5**
	CIFAR-10	**75**	**78.6**
	KWS	**100**	**93.1**

for experimental consistency to generate a KeyWord Spotting (KWS) dataset. Each dataset is split into training and test sets, as detailed below. We then assign training samples to $K = 100$ clients according to non-IID configurations and train $n = 10$ Convolutional Neural Networks (CNNs) as binary classifiers. By default, we select 20% of clients for training at each round, i.e., $C = 0.2$. Each client trains on its local dataset D_k for $E = 5$ epochs with a batch size of $B = 15$. The experimental settings for each dataset are as follows:

F-MNIST: F-MNIST consists of $60,000$ training samples and $10,000$ test samples. We employ the architecture in [11] using two convolution layers with 16 and 32 channels, respectively. Each convolution layer is followed by a 2×2 max-pooling layer and activated using the ReLU function.

CIFAR-10: For CIFAR-10, we adopt the convolutional VGG11 architecture [55] and distribute the $50,000$ training images to K clients for simulation.

KWS: We select 10 keywords (i.e., "one", "two", "three", "four", "five", "down", "left", "right", "stop", "go") from the Speech Commands dataset to generate our KWS dataset. We extract 50×16 Mel Frequency Cepstral Coefficients (MFCCs) as features by sampling each audio. We thereby obtain $21,452$ samples, of which $20,000$ are used for training and the rest for testing. We utilize a 3-hidden-layer CNN architecture with 16, 32, and 64 channels, followed by a fully-connected layer with 256 units. Each convolution layer is followed by a 1×2 max pooling layer. All convolution and fully-connected layers are ReLU-activated.

Remark: For the non-IID configuration, we use the parameter $1 \le l \le 10$ to indicate the number of unique labels held by the client. For instance, non-IID-2 means that each client has 2 distinct labels. This is achieved by grouping the training data by label and dividing each group into $(l \times K)/n$ partitions, finally assigning each client l partitions with different labels.

B. Performance Comparison with Different Distributed Optimization Algorithms

In this experiment, we compare the convergence speed of the proposed second-order optimization algorithm with the commonly used FedDANE [39] (Federated Newton-Type Method, second-order), FedAvg-based SGD (first-order), and FedAvg-based Adam (first-order) optimization algorithms. First, we conduct experiments on the F-MNIST, CIFAR-10, and KWS datasets under the IID setting to observe the convergence speed of different optimization algorithms. As shown in Table II, the proposed algorithm converges faster than the baseline methods on the F-MNIST, CIFAR-10, and KWS datasets under the IID setting. For example, For example, for the KWS dataset, our algorithm converges $2\times$ as fast as the popular Federal Newton method and $2.5\times$ faster than the conventional first-order optimization algorithm. The reason is that the gradient information and curvature information are used in the proposed algorithm to find a "better" descent direction to speed up the convergence speed.

However, as shown in Table II, it can be seen from the experimental results that although the proposed algorithm converges faster than the baseline methods, its performance is slightly lower than that of the baseline methods. This implies that convergence speed and accuracy are a trade-off in the distributed second-order optimization algorithm. In practice, SGD and Adam can use first-order information (i.e., gradient information) to find a better convergence point so that the performance is better than the second-order methods. It is worth noting that such a slight loss of precision in exchange for high communication efficiency is friendly to the resource-constrained FEEL framework. Therefore, in future research, we will focus more on improving the performance of the second-order optimization method.

C. Performance Comparison with Different Robust Training Schemes

First, we evaluate the performance of FedOVA with non-IID data against FedAvg. We adopt the same data distribution method for each dataset and send the training data to 100 clients according to the non-IID-2 configuration for experimental consistency. Our experimental results are shown in Fig.3. It is evident that FedOVA achieved higher classification accuracy than FedAvg on all three datasets. This is since FedOVA trains 10 binary classifiers asynchronously, each with high accuracy in distinguishing the current class from other classes. Moreover, the component binary classifiers in FedOVA are independent and have a low error correlation. They are also insensitive to missing samples in some classes. Therefore, an effective multi-class classifier can be constructed by ensembling these component classifiers.

To better investigate the performance of FedOVA under non-IID data distributions, we conduct a series of simulations with different non-IID configurations, i.e., we test non-IID-l for $l \in \{2, 3, 5\}$. Our simulation results are presented in Table III. Using either the FedOVA or the FedAvg algorithm, classification accuracy improved as the number of classes in the client's local dataset increased. However, there are still two points worthy of attention: the first is that different non-IID configurations had a dramatic impact on the performance of FedAvg but did not have a significant influence on FedOVA. The second is that the FedOVA achieved superior performance compared to FedAvg in every non-IID scenario. These two
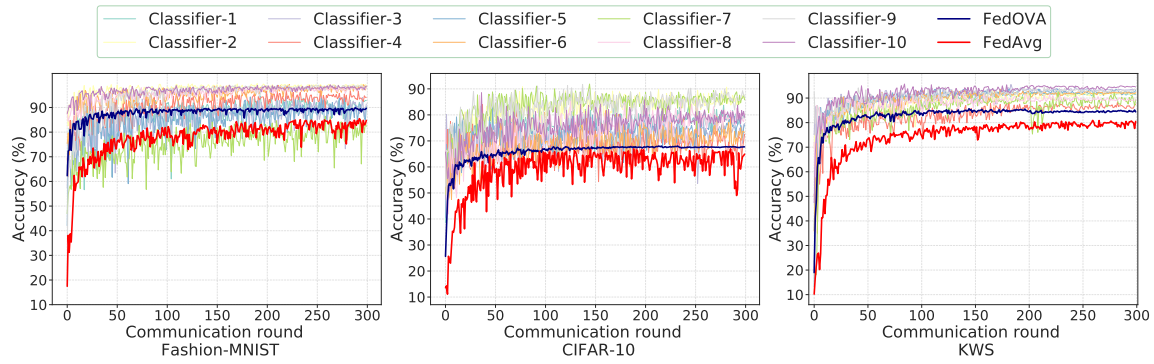
Fig. 3. Performance of FedOVA with non-IID data. Each client has samples from only two classes (non-IID-2).

TABLE III
ACCURACY (%) UNDER DIFFERENT NON-IID CONFIGURATIONS.

Dataset	F-MNIST			CIFAR-10			KWS		
Configuration	non-IID-2	non-IID-3	non-IID-5	non-IID-2	non-IID-3	non-IID-5	non-IID-2	non-IID-3	non-IID-5
FedAvg	84.3	85.8	89.9	63.5	66.3	72.5	80.5	82.4	86.0
FedOVA	**89.4**	**90.3**	**91.7**	**67.8**	**69.1**	**73.2**	**84.6**	**86.4**	**89.2**

TABLE IV
COMPARISON WITH [22] IN TERMS OF CLASSIFICATION ACCURACY (%) UNDER DIFFERENT DATA SHARING RATES β.

Dataset	F-MNIST	CIFAR-10	KWS
Data sharing ($\beta = 5\%$)	86.3	65.3	82.6
Data sharing ($\beta = 10\%$)	88.1	67.5	83.5
FedOVA	**89.4**	**67.8**	**84.6**

TABLE V
ACCURACY (%) VS. THE NUMBER OF CLIENTS K.

Dataset	F-MNIST		CIFAR-10		KWS	
K	100	1000	100	1000	100	500
FedAvg	84.3	83.1	63.5	61.2	80.5	75.9
FedOVA	**89.4**	**88.9**	**67.8**	**66.3**	**84.6**	**82.4**

observations fully illustrate the robustness of our proposed method for non-IID environments.

Second, we compare FedOVA with the data-sharing approach in [22], which is one of the most popular methods for performing federated training with non-IID data. We set the number of clients $K = 100$, $C = 0.2$, and use non-IID-2 as our experimental configuration. We define $\beta = D_{share}/D_k$ as the sharing rate, which represents the ratio of the data shared by the server to the local data. We randomly sample the global dataset to get the shared dataset D_s and send it to each client, where $D_s = D_k \times \beta$. We conduct simulations to compare the performance of the data-sharing strategy and FedOVA under different sharing rate β settings. As shown in Table IV, our approach still outperformed the data sharing strategy at a sharing rate of 5% and 10%. Only when the server has an extensive global dataset and shares more data with the client can the performance of data sharing outperform FedOVA. However, increased data sharing leads to a higher risk of privacy leakage, clearly not complying with privacy-preserving requirements. Furthermore, if β is too large, the generalization performance of the model is undermined. In contrast, our approach does not require sharing any data and provides complete privacy protection.

D. Effect of Large Scale Clients and Small Scale Samples

Considering the large number of clients that are commonly encountered in FEEL, we evaluate the performance of the FedOVA algorithm with a large-scale setting and compare it with that of FedAvg. We hold the total number of training samples constant and significantly increase the number of clients (i.e., K increases from 100 to 1000 in F-MNIST and CIFAR-10; since the amount of KWS data is limited in comparison, we set $K = 500$), so the amount of local data per client is reduced accordingly. For each round of training, we set $C = 0.2$ and select $K \times C$ clients to participate. We then set the average of the last 20 rounds as the final accuracy. The experimental results are summarized in Table V. It appears that the use of FedAvg led to a substantial decrease in accuracy as the number of clients increased; crucially, this was the case with FedOVA. Our method adopts the idea of OVA to train several binary classifiers, and the increase in the number of clients also improves the robustness of the binary classifier to different environments. In this way, it is still possible to achieve competitive results with fewer data.

E. Effect of Hyperparameter

In this subsection, we investigate the effect of different hyperparameter settings on classification accuracy to empirically validate the feasibility of our algorithm in cases that
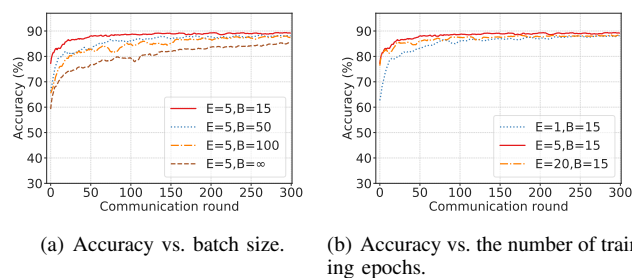
(a) Accuracy vs. batch size.   (b) Accuracy vs. the number of training epochs.

Fig. 4. Performance of FedOVA when varying either the number of epochs E or batch size B.

resemble real-world scenarios. We conduct experiments on the F-MNIST dataset to study the impact on classification accuracy and training convergence under different training epochs E and batch sizes B. For illustration purposes, we smoothen the curve and set the accuracy of the next 10 rounds as the value of the curve on each communication round.

One may observe in Fig. 4(a) that when fixing the number of training epochs $E = 5$, FedOVA achieved similar accuracy rates for different batch sizes ($B = 15, 50, 100, \infty$; note that ∞ means training on all the data in a single epoch). However, as the batch size in each epoch increased, the convergence of the algorithm slowed down. This result indicates that performing more gradient descent updates during each training round helps accelerate model convergence, enabling the model to reach a satisfactory performance in a short period of time. The results in Fig. 4(b) further verify this point. When we reduced the number training epochs, there was also a decrease in the convergence speed of the model in the same period of time. The above results are also in agreement with [11].

VII. CONCLUSION

This paper proposed a promising solution to simultaneously alleviate expensive communication costs and heterogeneous data problems for resource-constrained FEEL. Our insight is to design a robust training scheme for heterogeneous data and does not conflict with efficient communication algorithms. To improve the communication efficiency of resource-constrained FEEL, we have customized an FIM-based distributed approximate second-order optimization algorithm. Furthermore, we proved that the algorithm has linear convergence in the case of strong convexity and non-convexity, and analyzed its computational and communication complexity. Besides, to design an algorithm coupled with the above-mentioned communication efficient algorithm and robust to heterogeneous data, we first proposed a federated training scheme named FedOVA with the help of the traditional OVA method. With extensive experiments, we effectively validate the expected properties of our algorithm and empirically demonstrate its capability of reducing the communication cost and being robust to heterogeneous data.

In future work, we will further analyze the characteristics of each component classifier in FedOVA in an attempt to find an optimal training approach towards improving the performance of each model. Furthermore, we will investigate how to combine this scheme with asynchronous training to obtain better adaptation to real scenarios.

REFERENCES

[1] Y. Zhu, C. Markos, R. Zhao, Y. Zheng, and J. J. Yu, "Fedova: One-vs-all training method for federated learning with non-iid data," in *Proc. of IJCNN*, 2021, pp. 1–7.

[2] J. Kang, Z. Xiong, D. Niyato, Z. Cao, and A. Leshem, "Training task allocation in federated edge learning: A matching-theoretic approach," in *Proc. of WCNC*, 2020.

[3] J. Kang, Z. Xiong, C. Jiang, Y. Liu, S. Guo, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Scalable and communication-efficient decentralized federated edge learning with multi-blockchain framework," in *Proc. of BlockSys*. Springer, 2020, pp. 152–165.

[4] J. Ren, G. Yu, and G. Ding, "Accelerating dnn training in wireless federated edge learning systems," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 219–232, 2021.

[5] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "Hfel: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6535–6548, 2020.

[6] Y. Liu, R. Zhao, J. Kang, A. Yassine, D. Niyato, and J. Peng, "Towards communication-efficient and attack-resistant federated edge learning for industrial internet of things," *arXiv preprint arXiv:2012.04436*, 2020.

[7] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[8] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," *Proc. of INFOCOM*, 2020.

[9] J. Shu, W. Zhang, Y. Zhou, Z. Cheng, and L. T. Yang, "Flas: Computation and communication efficient federated learning via adaptive sampling," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2021.

[10] Y. Liu, X. Yuan, Z. Xiong, J. Kang, X. Wang, and D. Niyato, "Federated learning for 6g communications: Challenges, methods, and future directions," *China Communications*, vol. 17, no. 9, pp. 105–118, 2020.

[11] 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*. PMLR, 2017, pp. 1273–1282.

[12] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *Proc. of ICASSP*. IEEE, 2019, pp. 6341–6345.

[13] C. T. Dinh, N. H. Tran, T. D. Nguyen, W. Bao, and A. R. Balef, "Done: Distributed newton-type method for federated edge learning," *arXiv preprint arXiv:2012.05625*, 2020.

[14] W. Hao, D. N. Zakhary Kaplan, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. of INFOCOM*, 2020, pp. 1698–1707.

[15] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[16] Y. Zhu, S. Zhang, Y. Liu, D. Niyato, and J. J. Yu, "Robust federated learning approach for travel mode identification from non-iid gps trajectories," in *Proc. of ICPADS*, 2020, pp. 585–592.

[17] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *J. Mach. Learn. Res.*, vol. 5, pp. 101–141, Dec. 2004.

[18] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5g ultradense network," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2238–2251, 2021.

[19] Z. Ma, Y. Xu, H. Xu, Z. Meng, L. Huang, and Y. Xue, "Adaptive batch size for federated learning in resource-constrained edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[20] S. Prakash, S. Dhakal, M. R. Akdeniz, Y. Yona, S. Talwar, S. Avestimehr, and N. Himayat, "Coded computing for low-latency federated learning over wireless edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 233–250, 2021.

[21] X. Wang, R. Li, C. Wang, X. Li, T. Taleb, and V. C. M. Leung, "Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 154–169, 2021.

[22] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," *arXiv preprint arXiv:1806.00582*, 2018.

[23] M. Song, Z. Wang, Z. Zhang, Y. Song, Q. Wang, J. Ren, and H. Qi, "Analyzing user-level privacy attack against federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2430–2444, 2020.

[24] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2906–2920, 2019.

[25] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. Nguyen, and C. S. Hong, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 88–93, 2020.

[26] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained wireless federated learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 453–467, 2020.

[27] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," in *Proc. of AISTATS*. PMLR, 2021, pp. 2350–2358.

[28] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. of AISTATS*. PMLR, 2020, pp. 2021–2031.

[29] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. of ESORICS*. Springer, 2020, pp. 480–501.

[30] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *Proc. of RAID*, 2020, pp. 301–316.

[31] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *Proc. of ICLR*, 2019.

[32] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. of AISTATS*. PMLR, 2020, pp. 2938–2948.

[33] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. of ICML*. PMLR, 2019, pp. 634–643.

[34] L. Zhu and S. Han, "Deep leakage from gradients," in *Federated Learning*. Springer, 2020, pp. 17–31.

[35] F. Sattler, K.-R. M. Simon Wiedemann, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2020.

[36] Z. Zhang, W. Wu, J. Jiang, L. Yu, B. Cui, and C. Zhang, "Columnsgd: A column-oriented framework for distributed stochastic gradient descent," in *Proc. of ICDE*, 2020, pp. 1513–1524.

[37] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4583–4596, 2020.

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

[39] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smithy, "Feddane: A federated newton-type method," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2019, pp. 1227–1231.

[40] S. Wang, F. Roosta, P. Xu, and M. W. Mahoney, "Giant: Globally improved approximate newton method for distributed optimization," *Proc. of NeurIPS*, vol. 31, pp. 2332–2342, 2018.

[41] K. Kopparapu, E. Lin, and J. Zhao, "FedCD: Improving performance in non-IID federated learning," in *KDD Workshop on Artificial Intelligence of Things*, New York, NY, USA, 2020.

[42] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Journal of Machine Learning Research*, vol. 5, no. Jan, pp. 101–141, 2004.

[43] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[44] W. Chen, Z. Wang, and J. Zhou, "Large-scale l-bfgs using mapreduce," *Proc. of NeurIPS*, vol. 27, pp. 1332–1340, 2014.

[45] J. Liu, Y. Rong, M. Takác, and J. Huang, "On the acceleration of l-bfgs with second-order information and stochastic batches," *arXiv preprint arXiv:1807.05328*, 2018.

[46] C. Chen, S. Reiz, C. D. Yu, H.-J. Bungartz, and G. Biros, "Fast approximation of the gauss–newton hessian matrix for the multilayer

[47] J. Martens, "New insights and perspectives on the natural gradient method," *Journal of Machine Learning Research*, vol. 21, pp. 1–76, 2020.

[48] J. Friedman, T. Hastie, R. Tibshirani *et al.*, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1, no. 10.

[49] A. Ly, M. Marsman, J. Verhagen, R. P. Grasman, and E.-J. Wagenmakers, "A tutorial on fisher information," *Journal of Mathematical Psychology*, vol. 80, pp. 40–55, 2017.

[50] S. Hashemi, Z. M. Ying Yang, and M. Kangavari, "Adapted one-versus-all decision trees for data stream classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 5, pp. 624–637, 2009.

[51] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. of NeurIPS*, 2017.

[52] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[53] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *Technical Report, University of Toronto*, 2009.

[54] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[55] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of ICLR*, 2015.

[56] A. S. Berahas, J. Nocedal, and M. Takáč, "A multi-batch l-bfgs method for machine learning," in *Proc. of NeurIPS*, 2016, pp. 1063–1071.

perceptron," *SIAM Journal on Matrix Analysis and Applications*, vol. 42, no. 1, pp. 165–184, 2021.

Yi Liu (S'19) received the B.Eng degree from the Heilongjiang University, China, in 2019. His research interests include security and privacy in smart city and edge computing, deep learning, intelligent transportation systems, and federated learning.
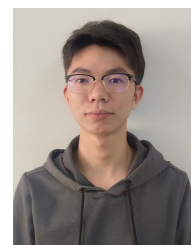
Yuanshao Zhu (S'20) received the B.Eng. degree in telecommunication engineering from Shandong University, Weihai, China, in 2019. He is currently a Master Student with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. His research interests include deep learning in smart city and edge computing, intelligent transportation systems, and federated learning.

James J. Q. Yu (S'11–M'15–SM'20) is an assistant professor at the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China, and an honorary assistant professor at the Department of Electrical and Electronic Engineering, the University of Hong Kong. He received the B.Eng. and Ph.D. degree in electrical and electronic engineering from the University of Hong Kong, Pokfulam, Hong Kong, in 2011 and 2015, respectively. He was a post-doctoral fellow at the University of Hong Kong from 2015 to 2018. He currently also serves as the chief research consultant of GWGrid Inc., Zhuhai, and Fano Labs, Hong Kong. His general research interests are in smart city and urban computing, deep learning, intelligent transportation systems, and smart energy systems. His work is now mainly on forecasting and decision making of future transportation systems and basic artificial intelligence techniques for industrial applications. He was ranked World's Top 2% Scientists by Stanford University in 2020. He is an Editor of the IET SMART CITIES journal and a Senior Member of IEEE.