FedOVA: One-vs-All Training Method for Federated Learning with Non-IID Data

Yuanshao Zhu,[1] Christos Markos,[1,2] Ruihui Zhao,[3] Yefeng Zheng,[3] and James J.Q. Yu[1,*]
[1]*Department of Computer Science and Engineering, Southern University of Science and Technology*
[2]*Faculty of Engineering and Information Technology, University of Technology Sydney*
[3]*Tencent Jarvis Lab*
yasozhu@gmail.com, christos.k.markos@gmail.com, zacharyzhao@tencent.com,
yefengzheng@tencent.com, *yujq3@sustech.edu.cn

***Abstract*—Federated Learning (FL) is a privacy-oriented framework that allows distributed edge devices to jointly train a shared global model without transmitting their sensed data to centralized servers. FL aims to balance the naturally conflicting objectives of obtaining massive amounts of data while protecting sensitive information. However, the data stored locally on each edge device are typically not independent and identically distributed (non-IID). Such data heterogeneity poses a severe statistical challenge for the optimization and convergence of the global model. In response to this issue, we propose Federated One-vs-All (FedOVA), an efficient FL algorithm that first decomposes a multi-class classification problem into more straightforward binary classification problems and then combines their respective outputs using ensemble learning. Experiments on several public datasets show that FedOVA achieves higher accuracy and faster convergence than federated averaging and data sharing. Furthermore, our approach can support practical settings with a large number of clients (up to 1000 clients) in FL.**

I. Introduction

Federated Learning (FL) [1] is a collaborative model training scheme that leverages large amounts of data distributed over multiple edge devices, without the latter sharing locally sensed data with any centralized entity. Concretely, edge devices (or *clients*) are asked to iteratively upload model updates to a centralized server, thereby jointly training a shared global model [2], [3]. On the other hand, the server is responsible for coordinating the distributed training process: it aggregates the local model updates uploaded by the clients in each training round, using them to optimize the global model. FL has spawned a series of emerging applications for on-device inference, such as Google Keyboard [4], smart voice assistants [5], and anomaly detection for time series data [6].

Recently, a communication-efficient model update aggregation algorithm, i.e., *Federated Averaging* (FedAvg) [1] (see Algorithm 1) was introduced to address the communication-related challenges in the FL framework. However, FL still faces statistical challenges. On the one hand, the FedAvg-based variant aggregation algorithms rely on Distributed Stochastic Gradient Descent (D-SGD) [7], which is widely used to iteratively train deep learning models under the setting of independent and identically distributed (IID) sampling of training data. The purpose of using the IID sampling method to learn from the training samples is to ensure that the stochastic gradient is an unbiased estimate of the full gradient [8], [9]. It is unrealistic to ensure that the local data on each edge client is always IID in practice. On the other hand, research indicates that data heterogeneity, i.e., non-IID distributions, may decelerate convergence [10], which is not conducive to obtaining a robust shared model.

Many factors can lead to the non-IID distribution, in this paper, we focus on addressing the problem of non-IID distribution caused by missing partial classes. There exist many works on mitigating the non-IID issue of FL, which address this issue by designing additional framework mechanisms. For example, a data sharing mechanism is designed in [7], [8] to improve FedAvg for non-IID data settings. It 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 to improve the performance of FL on non-IID data. [11] 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, which introduces additional computational overhead to the system. [12] proposed Favor, an experience-driven game framework that can intelligently select clients to participate in each round of training to balance the bias introduced by non-IID data and accelerate convergence. Nonetheless, in real-world applications, applying reinforcement learning within resource-constrained FL environments may be impractical.

Motivated by the above challenges, we aim to answer the following question: how to design a novel training algorithm

This work was supported by the General Program of Guangdong Basic and Applied Basic Research Foundation No. 2019A1515011032 and by the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation No. 2020B121201001. James J.Q. Yu is the corresponding author.

978-0-7381-3366-9/21/$31.00 ©2021 IEEE

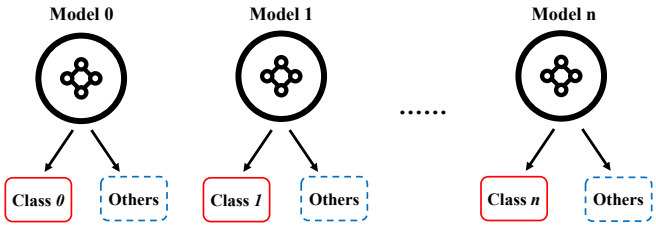


Fig. 1. The OVA method decomposes an n-class classification task into n binary classification tasks.

from the perspective of learning paradigm under the limitation of clients' non-IID data to achieve model performance comparable to that of the benchmark FL in [1] without bringing new problems to FL? Recall that, the non-IID problem in FL means that each client cannot obtain enough labels of multiple classes through data sharing to train a high-precision multiclass classification model, which limits the scalability of the FL framework. This implies that we cannot directly train a multiclass model on the client. To this end, our insight is to decompose a federated multiclass classification task on non-IID data into multiple binary classification tasks on the client-side by using the One-vs-All method. Therefore, in this paper, we design the Federated One-vs-All (FedOVA) algorithm by incorporating the idea of the One-vs-All method and the FedAvg training scheme. Such an algorithm can work effectively with the non-IID data in FL by decomposing the multi-classification task into multiple binary classification tasks.

In this work, we design a simple but elegant training algorithm FedOVA to address the non-IID problem in FL. Different from existing solutions, FedOVA addresses the non-IID problem from the perspective of learning. Specifically, the benefits that FedOVA brings to the framework are: (1) It improves the performance and convergence speed of the FL framework; (2) It does not bring additional operations to the FL framework. In addition, we conduct a series of comprehensive case studies on several public datasets (i.e., F-MNIST, CIFAR-10, and keyword spotting datasets) that empirically demonstrate the effectiveness of the proposed method. Specifically, FedOVA significantly outperforms FedAvg on established computer vision and natural language processing tasks, in both classification accuracy and convergence time.

II. Preliminaries

In this section, we provide an overview of the federated learning pipeline and the One-vs-All method.

A. Federated Learning Pipeline

In FL, we consider a server $\mathcal{S}$ and a set of clients $\mathcal{K}$ participating in training of a shared global model $F(\cdot)$. We assume that each client holds an IID or non-IID dataset D_k. Let $\ell(\omega; x)$ be the loss function at the client side for data sample x, where $\omega \in \mathbf{R}^d$ denotes the model's trainable

Algorithm 1 Federated Averaging (FedAvg) Algorithm

Server:

$\mathcal{K}$ is the client set
Initialize parameters ω^0
for each round $t = 1, \ldots, T$ **do**
  $\mathcal{K}_t \leftarrow$ (Sample a subset of clients from $\mathcal{K}$)
  for each client $k \in \mathcal{K}_t$ **in parallel do**
    $\omega_k^{t+1} \leftarrow \text{ClientUpdate}\ (k,\ \omega^t)$
  end for
  $\omega^{t+1} \leftarrow \frac{1}{|\mathcal{K}_t|} \sum_{k \in \mathcal{K}_t} \omega_k^{t+1}$
end for

ClientUpdate (k, ω):

η is the learning rate, D_k is the local dataset and $\nabla \ell(\cdot)$ is the local loss function.
for each epoch $e = 1, \ldots, E$ **do**
  $\mathcal{B} \leftarrow$ (split D_k into batches of size B)
  for batch $b \in \mathcal{B}$ **do**
    $\omega \leftarrow \omega - \eta \nabla \ell(\omega)$
  end for
end for
return ω to server

parameters. Thus, we let $\mathcal{L}(\omega) = E_{x \sim D}[\ell(\omega; x)]$ be the loss function at the server side. In particular, the client holds the data locally and only periodically exchanges updates with the server to optimize the following objective function:

$$\min_{\omega} \mathcal{L}(\omega), \text{ where } \mathcal{L}(\omega) := \sum_{k=1}^{K} p_k \mathcal{L}_k(\omega), \tag{1}$$

where K is total number of clients and p_k is a user-defined term indicating the relative influence of each client on the global model, with $\sum_k p_k = 1$.

To minimize the above objective function, the server and clients cooperatively run a T-round FL protocol. The main steps of this FL training protocol are follows:

- ***Step 1 (Initialization):*** In the t-th training round, the server randomly selects a subset of clients $\mathcal{K}_t$ from the client set $\mathcal{K}$ to participate in training, and sends them the initialized global model parameters ω^t.
- ***Step 2 (Local Training):*** Each client executes the $\text{ClientUpdate}(k, \omega^t)$ function to obtain the model updates. Using a local optimizer such as Adam, each client trains the received global model ω^t on the local dataset $\mathcal{D}_k$ and uploads the updates $\Delta\omega_k^t$ to the server.
- ***Step 3 (Aggregation):*** The server collects all updates uploaded by the clients and runs an aggregation algorithm (e.g., FedAvg [1], Algorithm 1) to obtain the optimized global model ω^*.

Note that the above steps are repeated in the same order until the global model converges.

B. One-vs-All Method

For an n-class classification task, we assume that it has a labeled dataset $D = \{x_i, y_i\}_{i=1}^N$ where y_i is the label of x_i

and $y_i \in \{1, 2, \ldots, n\}$, N is the total number of samples. As shown in Figure 1, the OVA method needs to train n binary classifiers $f_i(\cdot), i \in \{1, 2, \ldots, n\}$, and each classifier discriminates one class from others on the dataset D. Each classifier can be expressed as follows:

$$f_i(x) = P(y = i|x; \omega_i), \tag{2}$$

where ω_i is the parameters of classifier $f_i(\cdot)$. When we need to request a prediction of a new instance, OVA gives us result by using these n binary classifiers to obtain the confidence that the instance belongs to the current class. Specifically, we select the one with the highest confidence as the final prediction result for this new instance. Such a method is formalized as follows:

$$\hat{y} = \underset{i \in \{1,2,\ldots,n\}}{\arg\max} f_i(x). \tag{3}$$

C. Discussion

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 FL to propose a novel algorithm, called FedOVA. This algorithm is able to solve the non-IID data problem in FL efficiently, which will be described in Section III. To the best of our knowledge, this is the first work that integrates FL and OVA to address FL with non-IID data.

However, there are some essential differences between centralized learning and federated learning which raise several design challenges: (1) Different from the server in centralized learning that can access the raw data, the server in federated learning cannot access the raw data of the client due to privacy protection requirements, and thus cannot distribute appropriate classifiers f_i to the client; (2) Due to the limitation of the client's computing resources, when introducing the OVA method, we should not impose too much additional computing burden on the client and server.

III. Federated One-vs-All Algorithm

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

A. Overview of FedOVA Algorithm

We assume that there is a server responsible for coordination and a client set $\mathcal{K}$ in FL where each client has a local dataset $D_k, k \in \{1, 2, \ldots, K\}$ and the total number of classes is n. The training process of FedOVA is to repeat the following steps for communication round 1 to T:

- ***Step 1 (Initialization):*** For each round of training, the server randomly selects a subset $\mathcal{K}_t$ of client set $\mathcal{K}$ to participate in FL training with the fraction C. Then, the server broadcasts the binary classifier model parameters ω_i^t to these clients, where i denotes the classifier ID, $i \in \{1, 2, \ldots, n\}$.
- ***Step 2 (Local Training):*** After receiving the binary classifier model parameters ω_i^t, the client initializes some of the OVA component classifier models according to its own local data label distribution, i.e., $\omega_{i,k}^t \leftarrow \omega_i^t$. For example, for F-MNIST dataset, if client k only has label "1" and label "2", then this client initializes parameters of classifiers f_1 and f_2. For each binary classifier, the goal is to minimize the following objective function:

$$\arg\min_{\omega} \mathcal{L}(\omega) = \frac{1}{|D_k|} \sum_{\{x_i, y_i\} \in D_k} \ell(y_i, f_i(x_i; \omega)), \tag{4}$$

  where D_k denotes the local dataset that contains training samples (x_i, y_i), ω is the parameter of binary classifier, and $\ell(\cdot, \cdot)$ is a loss function (i.e., $\ell(y_i, f(x_i; \omega)) = \frac{1}{2}({x_i}^T \omega - y_i)$). Each client performs stochastic gradient descent to optimize the classifier by using the local dataset and then sends the updates back to the server.
- ***Step 3 (Aggregation):*** Since each client trains only some of the classifiers, not returning parameters for all of them, and each classifier is independent of each other, we can perform asynchronous updates to reduce the computational burden. The server groups returned parameters according to their corresponding binary classifier models and then aggregates the parameters of each group G_i. For model f_i, the aggregation process can be formulated as:

$$\omega_i^{t+1} = \frac{1}{|G_i^t|} \sum_{k \in G_i^t} \omega_{i,k}^t. \tag{5}$$

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

FedOVA aims to train an expert binary classifier for each class, hence solving the problem of difficulty in the convergence of training multi-classifier models with non-IID data.

B. Strengths of the FedOVA Algorithm

Conventional OVA combined with FL can be used to solve non-IID data problems due to the following advantages of FedOVA:

- **Independent classifiers.** FedOVA is based on the idea of ensemble learning to combine a series of classifiers. Unlike the multi-class classifier ensemble, each classifier of FedOVA is specialized in distinguishing a specific class. such a design can ensure low error correlation among different classifiers, thus enhancing the diversity among classifiers and improving the overall classification accuracy [13].
- **Competitive classification accuracy.** As an ensemble learning scheme, FedOVA is capable of achieving high classification accuracy. [14], [15] has shown that when its binary classifiers are well tuned, OVA can achieve

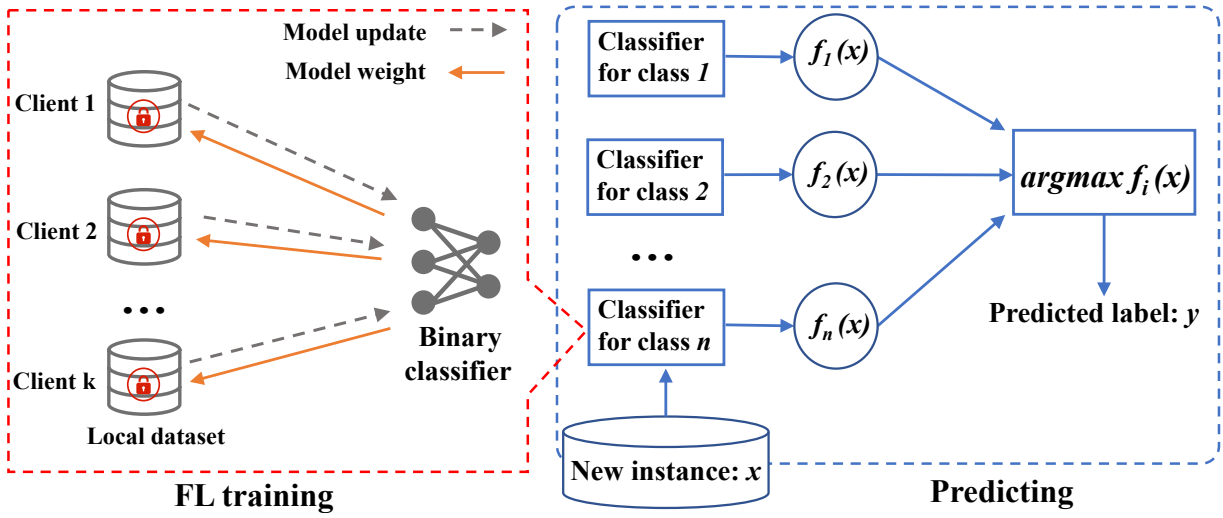


Fig. 2. Overview of the FedOVA algorithm, which trains n binary classifiers and selects the output of the most confident classifier as the prediction result.

Algorithm 2 Federated One-vs-All (FedOVA) Algorithm

Server:

$\mathcal{K}$ is the client set
Initialize n component models $\{f_1, \ldots, f_n\}$
Initialize parameters of all component models ω_m^0
for each round $t = 1, \ldots, T$ **do**
  $\mathcal{K}_t \leftarrow$ (Sample a subset of clients from $\mathcal{K}$)
  Send the parameters w_i^t, $i \in \{1, 2, \ldots, n\}$ to $\mathcal{K}_t$
  for each client $k \in \mathcal{K}_t$ **do**
    ClientUpdate (k, ω_i^t)
  end for
  for each group of component model $f_i \in \{f_1, \ldots, f_n\}$ **do**
    $\omega_i^{t+1} \leftarrow \frac{1}{|G_i^t|} \sum_{k \in G_i^t} \omega_{i,k}^t$
  end for
end for

Client:

for each component model $f_i \in \{f_1, \ldots, f_n\}$ **do**
  Update component model parameters ω_i based on its local data
end for
return ω_i to server

accuracy on par with any other multi-classification approaches.

- **No requirement for all sample classes.** Since each component classifier in FedOVA assigns samples that do not belong to the current class to others, missing negative sample classes does not significantly affect the classifier performance [16]. Besides, when new label classes emerge, FedOVA only needs to create a new classifier for each new emergent label, which allows adaptation to new environments without drastic changes during FL training.
- **Asynchronous updates.** There is no redundancy among the component binary classifiers in FedOVA, each is trained without affecting the other. Therefore, during the FL training process, component binary classifiers can be independently updated once they have finished training, thus increasing client-server communication efficiency.

IV. Experiments Evaluation

In this section, we conduct simulation experiments on three representative public datasets to validate the performance of FedOVA under non-IID data setting. All simulation experiments were developed using Python 3.7 and PyTorch 1.7 [17] and run with an Nvidia GeForce RTX2080 Ti GPU and an Intel® Xeon® Silver CPU. All experiments are executed sequentially to mimic distributed training.

A. Experimental Setup

We conduct extensive experiments on three established datasets, namely F-MNIST [18], CIFAR-10 [19] and the Speech Commands dataset [20]. The first two are image datasets with 10 classes and have been widely adopted for FL benchmarks. The Speech Commands dataset contains 35 classes, where each sample is a 1-second long audio stored in the WAV format. For consistency, we select 10 commonly used keywords to generate a KeyWord Spotting (KWS) dataset. Each dataset is split into training and test sets. We 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 5 epochs with a batch size of 15 (i.e., $E = 5$ and $B = 15$). The specific experimental settings of the dataset are as follows:

- **F-MNIST:** F-MNIST consists of $60,000$ training samples and $10,000$ test samples. We employ an architecture in

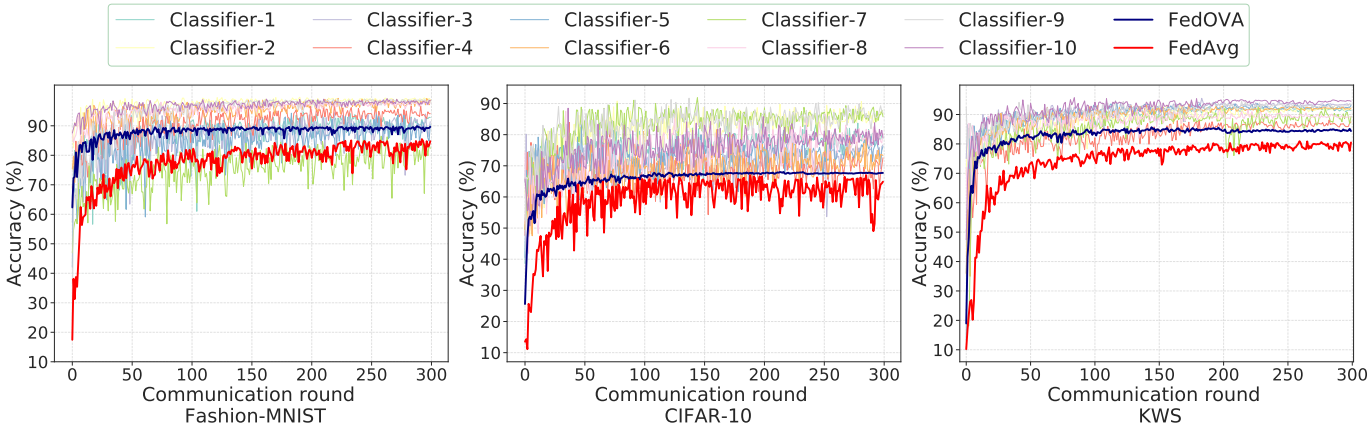


Fig. 3. Performance of FedOVA with non-IID data, each client has samples from only two classes, non-IID-2.

TABLE I
ACCURACY COMPARISON 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**

[1] using two convolutional layers with 16 and 32 channels, respectively. Each convolutional layer is followed by a 2×2 max pooling layer and activated using the ReLU function.

- **CIFAR-10:** We adopt a popular CNN architecture VGG11 [21] on CIFAR-10 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 as a 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-layer CNN architecture with 16, 32, and 64 channels, respectively. Each convolutional layer is followed by a 1×2 max pooling layer. Finally, there is a fully-connected layer with 256 units. All convolution and fully-connected layers are ReLU-activated.

Remark: All of our deep learning architectures here are not aimed at the state-of-the-art results, as our goal is to evaluate our algorithms rather than obtain the best accuracy on every task.

For the non-IID configuration, we use the parameter $1 \leq l \leq 10$ 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 l partitions with different labels to each client.

B. FedOVA Performance

First, we evaluate the performance of FedOVA with non-IID data against FedAvg. For experimental consistency, 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. Our experimental results are shown in Figure 3. We can clearly observe that FedOVA is able to achieve higher accuracy than FedAvg on all dataset tasks, and is able to achieve faster and more stable convergence. This result shows that FedOVA significantly outperforms FedAvg in the same scenario, which also demonstrates FedOVA's excellent performance in the context of non-IID data distribution. Meanwhile, Figure 3 also provides a strong explanation for such good performance of FedOVA. FedOVA trains 10 binary classifiers asynchronously, each with high accuracy in distinguishing the current class from other classes. Moreover, the internal binary classifiers in FedOVA are independent of each other and have low error correlation, and the binary classifiers are insensitive to missing samples in some classes. Therefore, a multi-classifier robust to non-IID scenarios can be constructed by ensemble 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 $l \in \{2, 3, 5\}$. The simulation results are demonstrated in Table I. Whether using the FedOVA or FedAvg algorithm, the accuracy improve as the number of label classes in the client's local dataset increases. However, there are still two points worth of attention: The first is that different non-IID configurations

TABLE II
COMPARE TO THE METHOD [8] WITH DIFFERENT DATA SHARING RATE (β). ACCURACY IS REPORTED IN PERCENTAGE (%).

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**

have a dramatic impact on the performance of the FedAvg algorithm but do not have a significant influence on FedOVA. The second is that the FedOVA is also able to achieve superior performance compared to FedAvg in each non-IID scenario. These two observations fully illustrate the robustness of our method for non-IID environments.

C. FedOVA vs. Data Sharing

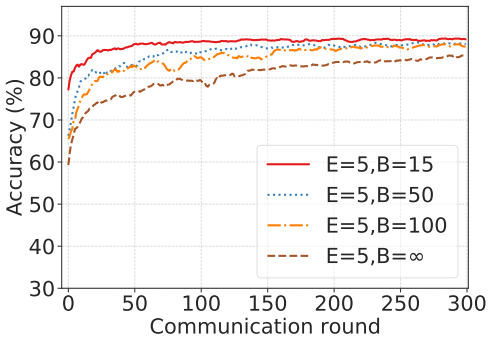
In this subsection, we compare our approach with the data sharing approach in [8], which is one of the most popular methods for performing federated learning with non-IID data. We set the number of clients $K = 100$, $C = 0.2$, and use non-IID-2 as our experimental configuration. We define $\beta = D_{share}/D_k$ 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 $D_s = D_k \times \beta$. Then, we conduct simulation experiments to compare the performance of the data sharing strategy and FedOVA under different sharing rate β settings. As shown in Table II, our approach still outperforms the data sharing strategy at a sharing rate of 5% and 10%. Only when the server has a large global dataset and shares more data with the client, the performance of data sharing strategy can outperform FedOVA. However, the increased data sharing leads to a higher risk of privacy leakage, clearly not in compliance with the 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. Large Scale Clients and Small Scale Samples

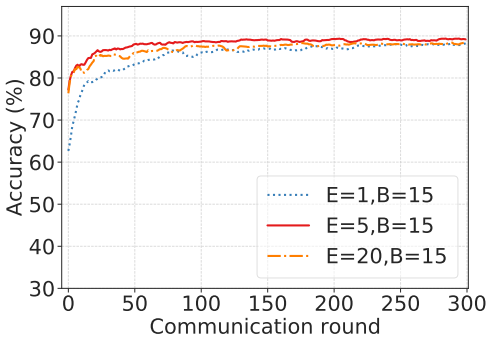
Considering the large number of clients that are common in FL, we evaluate the performance of the FedOVA algorithm with a large-scale setting and compare it with FedAvg. We hold the total number of training samples constant and increase the number of clients significantly (i.e., K increases from 100 to 1000 in F-MNIST and CIFAR-10; since the amount of KWS data is too limited, we set $K = 500$), so the amount of local data per client will be 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 III. It appears that the use of FedAvg leads to a substantial decrease in accuracy as the number of clients increases; crucially, this is not the case with FedOVA. The reason is that our method adopts the idea of OVA to train a number of binary classifiers and the increase in the number of clients also increases the robustness of the binary classifier to different environments. In this way, it is still possible to achieve a competitive results with fewer data.

TABLE III
ACCURACY VS. 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**



(a) Accuracy comparison with different batch size settings.



(b) Accuracy vs. the number of training epochs.

Fig. 4. Performance of FedOVA with different settings of the epochs (E) and batch size (B).

E. Hyperparameter Analysis

In this subsection, we investigate the effect of different hyperparameter settings to empirically validate the feasibility of our algorithm in cases that resembles real-world scenarios. We conduct experiments on the F-MNIST dataset, mainly to study the impact on the accuracy and convergence of the algorithm with different training epochs E of the client and the batch size 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.

We can see from Figure 4(a) that when fixing the same training epoch ($E = 5$) for the client, the algorithm can achieve a similar accuracy rate with different batch size ($B = 15$, 50, 100, ∞; ∞ means training with all the data on the client in one epoch). However, as the batch size in each

epoch gradually increases the convergence of the algorithm will slow down. This result indicates that performing more gradient descent updates during each training round helps improve the convergence speed of the model, enabling the model to reach a satisfactory performance in a short period of time. The results in Figure 4(b) further verify this point. When we reduce the training epoch, there is also a decrease in the convergence speed of the model in the same period of time. The above results are also in agreement with [1].

However, we can not use a large epoch and a small batch size without limitations, which increase the computational burden of the edge devices. In addition, a large number of epoch (i.e., the client does not communicate with the server for a long time) can increase the risk of the local model be attacked.

V. Conclusion

In this paper, we propose FedOVA, an OVA-based FL training algorithm that significantly improves the performance of FL in non-IID data scenarios. In our extensive experiments on established computer vision and speech recognition datasets, FedOVA consistently outperforms FedAvg under various non-IID configurations. Our results also show that FedOVA converges faster with less fluctuation in accuracy, thereby reducing the negative impact of non-IID data on model convergence. Finally, our experimental results suggested that FedOVA can be applied to large-scale user scenarios, especially in the case of a large number of clients (up to 1000 clients) with small amounts of data each.

A limitation to FedOVA is that it will not be possible to train binary classifiers when each client has only one type of label, since training each binary classifier requires both positive and negative samples. In fact, all existing methods will perform poorly in this extreme context.

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

References

[1] 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.

[2] P. Abeysekara, H. Dong, and A. K. Qin, "Distributed machine learning for predictive analytics in mobile edge computing based iot environments," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.

[3] S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang, "Learning private neural language modeling with attentive aggregation," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[4] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[5] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2019, pp. 6341–6345.

[6] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial IOT: A communication-efficient on-device federated learning approach," *IEEE Internet of Things Journal*, 2020.

[7] 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.

[8] 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.

[9] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–9.

[10] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *International Conference on Learning Representations*, 2020.

[11] 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.

[12] W. Hao, D. N. Zakhary Kaplan, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *IEEE Conference on Computer Communications*, 2020, pp. 1698–1707.

[13] 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.

[14] 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.

[15] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, 2011.

[16] Z. Zhang, B. Krawczyk, S. Garcia, A. Rosales-Perez, and F. Herrera, "Combining one-vs-one decomposition and ensemble learning for multi-class imbalanced data," *Knowledge-Based Systems*, vol. 106, no. Aug.15, pp. 251–263, 2016.

[17] 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 *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, 2017.

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

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

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

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.