

# Local Data Quantity-Aware Weighted Averaging for Federated Learning with Dishonest Clients

Leming Wu<sup>1</sup>, Yaochu Jin<sup>1,2,\*</sup>, Kuangrong Hao<sup>1</sup>, Han Yu<sup>3</sup>

<sup>1</sup>College of Information Science and Technology, Donghua University, Shanghai, China

<sup>2</sup>School of Engineering, Westlake University, Hangzhou, China

<sup>3</sup>College of Computing and Data Science, Nanyang Technological University (NTU), Singapore

lemingwu@mail.dhu.edu.cn, jinyaochu@westlake.edu.cn, krhao@dhu.edu.cn, han.yu@ntu.edu.sg

**Abstract**—Federated learning (FL) enables collaborative training of deep learning models without requiring data to leave local clients, thereby preserving client privacy. The aggregation process on the server plays a critical role in the performance of the resulting FL model. The most commonly used aggregation method is weighted averaging based on the amount of data from each client, which is thought to reflect each client’s contribution. However, this method is prone to model bias, as dishonest clients might report inaccurate training data volumes to the server, which is hard to verify. To address this issue, we propose a novel secure Federated Data quantity-aware weighted averaging method (FedDua). It enables FL servers to accurately predict the amount of training data from each client based on their local model gradients uploaded. Furthermore, it can be seamlessly integrated into any FL algorithms that involve server-side model aggregation. Extensive experiments on three benchmarking datasets demonstrate that FedDua improves the global model performance by an average of 3.17% compared to four popular FL aggregation methods in the presence of inaccurate client data volume declarations.

**Index Terms**—Federated learning, Privacy preserving, Aggregation weights, Collaborative training

## I. INTRODUCTION

In recent years, artificial intelligence (AI) technology has made significant progress and is now integrated into many aspects of daily life, including intelligent question-answering systems [1], smart finance [2], and autonomous driving [3], etc. While the development of AI has greatly benefited society by providing convenience, it has also raised concerns about the leakage of user privacy data, as technology can act as a double-edged sword. Federated learning (FL) [4]–[6], which allows deep learning models to be trained without transferring local data, can effectively mitigate the risk of privacy breaches.

FL can be classified into centralized FL [7], [8], which involves a central node server, and decentralized FL [9], which does not. Additionally, based on the relationship between data labels and features, FL can be categorized into horizontal [10], [11] and vertical types [12]. This paper primarily focuses on centralized FL with a central node server. In this setup, the server typically performs weighted aggregation of model parameters. As shown in Eq. (2), the most common method of weighted aggregation is based on the amount of client

data. This approach has several advantages in FL. It better reflects the contribution of data to the global model, improves the model’s generalization ability, and reduces the impact of noise from clients with small sample sizes, while balancing fairness and efficiency. In addition, this method is simple, easy to implement, and easy to interpret.

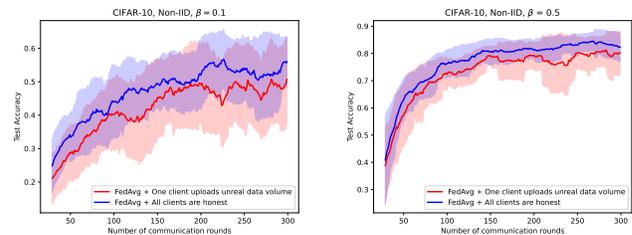


Fig. 1. This experiment evaluates the global model accuracy of the FedAvg algorithm on the CIFAR-10 dataset, comparing scenarios with and without dishonest clients manipulating data volume. Two levels of Non-IID data distribution, defined by Dirichlet parameters  $\beta = 0.1$  and  $\beta = 0.5$ , are considered.

When performing weighted aggregation based on the amount of client data, the server can only aggregate according to the data reported by the clients. Dishonest clients can manipulate their local model weights by providing false data amounts. We experimentally verified this behavior using the classic FL algorithm, FedAvg. In each round, 10 clients were randomly selected from a total of 100 to participate in federated training. If one client dishonestly reports its data amount, we observed a significant drop in the global model’s accuracy, as shown in Fig. 1.

To address this issue, we propose a first-of-its-kind secure Federated Data quantity-aware weighted averaging method (FedDua). The core idea is to design a data quantity-aware branch and integrated it into the client model. This branch accurately predicts the adjustment factor  $\alpha$  related to the amount of data. By combining  $\Delta\theta$  from the client model, the learning rate, and the average gradient from client training, we can estimate the client’s data amount. Additionally, we found that the distribution of the adjustment factor  $\alpha$  remains similar across different amounts of data. Therefore, once the

\*Corresponding author.

server receives the adjustment factor  $\alpha$  from the client, it can verify the reported data amount by comparing it with the pre-trained distribution of  $\alpha$ . If a dishonest client is detected, the server will issue a warning, encouraging the client to report the data amount honestly. If the client persists in dishonesty, the server can exclude it from federated training.

Our proposed FedDua method can be implemented as a module in other federated algorithms that require aggregation. Extensive experiments have demonstrated that the algorithm can successfully identify clients that dishonestly report data amounts and mitigate the reduction in model accuracy caused by such dishonest reporting. Extensive experiments on three benchmarking datasets demonstrate that FedDua improves the global model performance by an average of 3.17% compared to four popular FL aggregation methods in the presence of inaccurate client data volume declarations.

## II. RELATED WORK

Existing methods [7], [13]–[15] for weighted aggregation in FL typically perform aggregation based on the amount of client data. This approach enhances the stability and generalization performance of the global model, making it an important and effective aggregation strategy. However, if a dishonest client uploads an incorrect amount of training data, the global model’s performance will degrade. To address this, [16], [17] proposed a weighted aggregation method based on client contributions. However, methods that more accurately measure client contributions, such as those in [18], [19], often involve high computational complexity. [20] proposed using the client’s verification loss as a measure of contribution, allowing weighted aggregation based on verification loss. However, this method is sensitive to data volume differences and Non-IID data, and it introduces additional computational overhead and instability. [21], [22] proposed robust aggregation strategies, but these still rely on client data volume for weighted aggregation.

While these methods improve the robustness of FL to some extent, they significantly reduce training performance when dishonest clients are present. The proposed FedDua approach is designed to bridge this important gap in current FL literature.

## III. PRELIMINARIES

In an FL scenario with a central server, there are  $N$  clients, each with data  $D_i = (x_i, y_i), i = \{1, \dots, |D_i|\}$ , where  $|D_i|$  denotes the number of data points on client  $i$ , and  $(x_i, y_i)$  represents the input sample  $x_i$  and its corresponding label  $y_i$ . In each communication round, the server selects  $K$  clients to participate in federated training. Initially, the server sends the global model  $\theta_g^0$  to each client. Upon receiving the model, client  $i$  trains its local model  $\theta_i^0$  using its local data  $D_i$  without transferring data, as shown in Eq. (1). Here,  $f(x_i; \theta)$  is the model’s prediction, and  $l$  is the loss function. The client then uploads its model parameters  $\theta_i^1$  to the server, where the parameters are aggregated, as shown in Eq. (2).

$$\mathcal{L}_i(\theta_i; D_i) = \mathbb{E}_{(x_i, y_i) \sim D_i} [l(f(x_i; \theta), y_i)], \quad (1)$$

$$\min_{\{\{\theta_i\}\}_{i=1}^K} \frac{|D_i|}{\sum_{i=1}^K |D_i|} \sum_{i=1}^K \mathcal{L}_i(\theta_i; D_i). \quad (2)$$

If there is an incorrect data volume in the weighted aggregation of Eq. (2), the actual data volume is denoted as  $|D_j|_{real}$ , and the reported (false) data volume is  $|D_j|_{unreal}$ , where  $|D_j|_{real} \neq |D_j|_{unreal}$ . The weight in the global loss function is determined by the data volume. If client  $j$  dishonestly reports its data volume, the optimization objective becomes:

$$\mathcal{L}' = \sum_{i \neq j} w_i \cdot \mathcal{L}_i + (w_j + \Delta w_j) \cdot \mathcal{L}_j. \quad (3)$$

Here,  $\Delta w_j = w'_j - w_j$ , which shows that if  $\Delta w_i > 0$ , the increase in  $w_j + \Delta w_j$  amplifies the impact of client  $j$  on the global model. Additionally, if the loss distribution  $\mathcal{L}_j$  differs significantly from that of other clients, the global optimization direction will be biased towards client  $j$ .

---

### Algorithm 1: The pseudo-code of FedDua

---

```

1 for each round of local model update do
2   S ← The set of clients;
3   for each client  $i$  in the set of clients  $S$  do
4      $\theta_i^t \leftarrow$  Train the local model by the local data in
        $i$ -th client;
5      $\nabla \theta_i^t \leftarrow$  Calculate the model’s gradient in the
       local training;
6      $\alpha_i^t \leftarrow$  Calculate the  $\alpha$  by training data
       quantity-aware branch;
7     Send  $(\nabla \theta_i^t, \alpha_i^t)$  to the server;
8   end
9   On the server:
10  Verify  $\alpha$  and the amount of data on the client
       through  $\alpha, \nabla \theta_i^t$  and  $\Delta \theta$ ;
11  The server evaluates the reliability of the data
       provided by each client by combining  $\alpha_i^t$  with the
       pre-trained distribution of  $\alpha$ ;
12  if  $\alpha_i^t$  is not True then
13     $|D_i| \leftarrow$  For clients that are found to upload
       unrealistic amounts of data, aggregate based
       on the predicted amount of data:
        $|D_i| = \frac{\Delta \theta_i}{\eta \cdot \nabla \theta_i \cdot \alpha}$ ;
14  else
15     $\theta_g^t \leftarrow$  Aggregate based on the amount of
       training data uploaded by the client:
        $\theta_g^t = \sum_{i=1}^K \frac{|D_i|}{\sum_{i=1}^K |D_i|} \theta_i^t$ ;
16  end
17  Send  $\theta_g^t$  to clients in set  $S$ ;
18 end

```

---

## IV. THE PROPOSED FedDua APPROACH

In this section, we describe the proposed algorithm FedDua. The algorithm consists of two main parts: the client obtains the adjustment factor  $\alpha$  through the quantity-aware branch, and the server determines whether the data volume

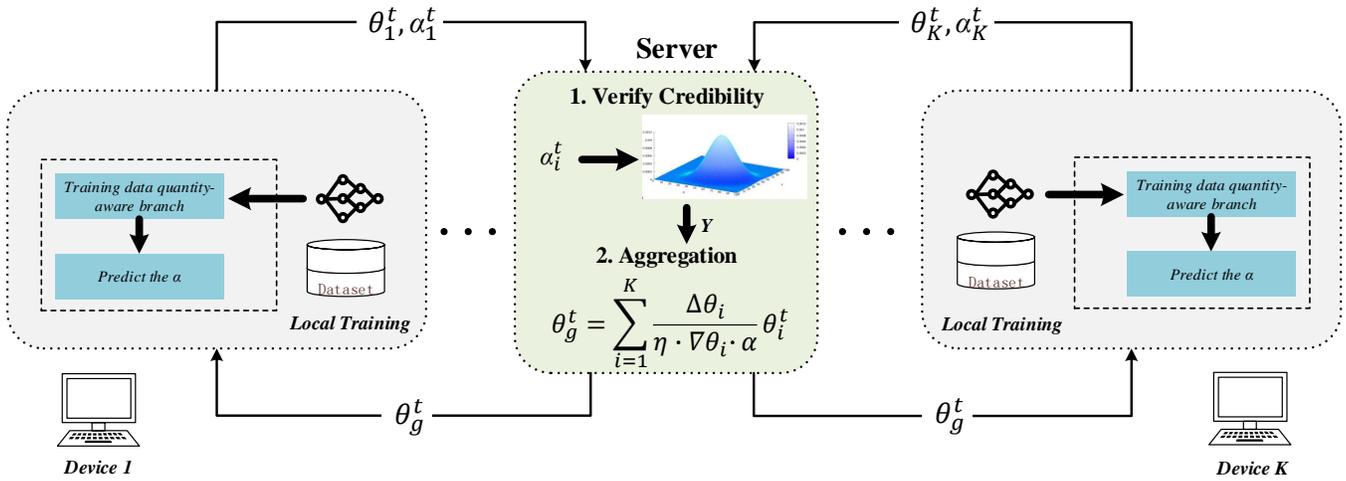


Fig. 2. The proposed FedDua approach architecture.

uploaded by the client is credible. The architecture of the proposed algorithm is shown in Fig. 2. The detailed process of the algorithm is provided in Algorithm 1.

#### A. Quantity-aware Branch

If the client directly uploads the number of training data to the server, the server cannot verify its authenticity. To address this, we designed a quantity-aware branch and integrated it into the client model. This branch enables accurate prediction of the client's training data volume based on the model parameter change  $\Delta\theta$ , the learning rate  $\eta$ , and the average gradient of the mini-batch.

During the training of the neural network model, the training set is divided into multiple batches. The model is updated once for each batch, as shown in Equations 4 and 5, where  $\nabla L_r(\theta_i)$  represents the model gradient at the  $r$ -th iteration. One epoch of training requires  $R$  iteration updates. Therefore, the client's training data volume is  $R \times \text{batch\_size}$ . From Eq. (5), we can deduce  $R \approx \frac{\Delta\theta_i}{\eta \cdot \mathbb{E}[\nabla L(\theta_i)]}$ , where  $\Delta\theta_i$  is the model parameter difference for one epoch. However, since both  $\Delta\theta_i$  and  $\nabla L(\theta_i)$  are vectors rather than scalars, directly taking the norm for calculation may cause significant deviation. To address this, we use the quantity-aware branch to predict the adjustment factor  $\alpha$ . Using  $\alpha$ , we can accurately estimate the client's training data volume, as shown in Eq. (6).

$$\theta' = \theta - \eta \nabla L_r(\theta), \quad (4)$$

$$\theta^{(t+1)} = \theta^t - \eta \sum_{r=1}^T \nabla L_r(\theta), \quad (5)$$

$$R = \frac{\Delta\theta_i}{\eta \cdot \mathbb{E}[\nabla L(\theta_i)] \cdot \alpha}. \quad (6)$$

The client deploys the quantity-aware branch to predict the  $\alpha$  value, as shown in Eq. (7). Here,  $\varphi$  is the parameter of the quantity-aware branch, and  $\text{embedding}(\text{client}_i)$  is the input to the quantity-aware branch corresponding to client  $i$ , which can

either be fixed or learnable. The loss function of the quantity-aware branch is shown in Eq. (8). Using this loss function, we can train the quantity-aware branch  $f_{dua}$  so that its predicted  $\alpha$  accurately estimates the client's training data volume.

$$\alpha = f_{dua}(\varphi; \text{embedding}(\text{client}_i)), \quad (7)$$

$$\text{Loss}_{dua} = \frac{1}{2} \left\| \frac{\Delta\theta_i}{\eta \cdot \mathbb{E}[\nabla L(\theta_i)] \cdot \alpha} - |D_i| \right\|^2. \quad (8)$$

The optimization process of the quantity-aware branch is as follows: the gradient of  $\text{Loss}_{dua}$  with respect to  $\alpha$  is shown in Eq. (9), and the chain rule for the derivative of the loss  $\text{Loss}_{dua}$  with respect to  $\varphi$  is shown in Eq. (10). Finally, the parameter  $\varphi$  of the quantity-aware branch can be updated using Eq. (11).

$$\frac{\partial \text{Loss}_{dua}}{\partial \alpha} = \left( \frac{\Delta\theta_i}{\eta \cdot \mathbb{E}[\nabla L(\theta_i)] \cdot \alpha} - |D_i| \right) \cdot \left( -\frac{\Delta\theta_i}{\eta \cdot \mathbb{E}[\nabla L(\theta_i)] \cdot \alpha^2} \right), \quad (9)$$

$$\frac{\partial \text{Loss}_{pred}}{\partial \varphi} = \frac{\partial \text{Loss}_{pred}}{\partial \alpha} \cdot \frac{\partial \alpha}{\partial \varphi}, \quad (10)$$

$$\varphi^{(t+1)} = \varphi^{(t)} - \lambda \cdot \frac{\partial \text{Loss}_{pred}}{\partial \varphi}. \quad (11)$$

In FL, the client uploads the model gradient parameters to the server in each communication round, and the server aggregates the model gradients from the clients. During each communication round, the client typically performs  $E$  epochs of training. After each epoch, the client predicts the adjustment factor  $\alpha$  using the quantity-aware branch. At the end of the communication round, the client uploads the  $E$  adjustment factors  $\alpha$  and the parameter  $\varphi$  of the quantity-aware branch to the server.

After receiving the model parameters uploaded by the client, the server will first verify whether the amount of training data

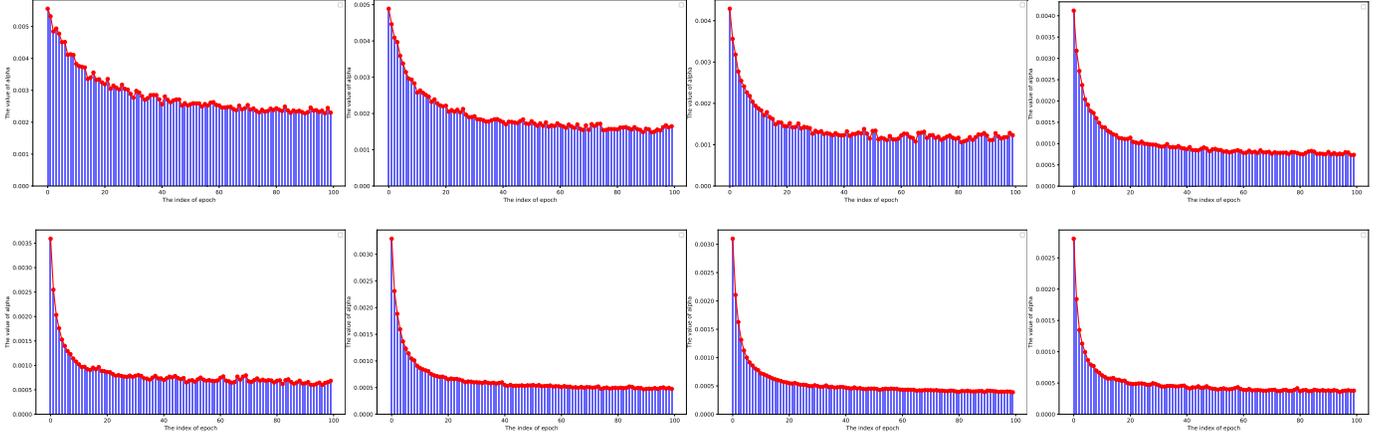


Fig. 3. Distribution of  $\alpha$  values under different communication rounds using data quantity-aware branch prediction by the client under different data amounts.

reported by the client is accurate and reliable, based on the adjustment factor. For the specific verification process, refer to part IV-B. Once the server confirms that the reported data volume is accurate, it will perform weighted aggregation on the model parameters  $\theta$  and the parameters  $\varphi$  of the quantity-aware branch, based on the client’s actual data volume.

Compared to directly calculating the adjustment factor  $\alpha$  using Eq. (12), using the quantity-aware branch to predict  $\alpha$  offers three main advantages. First, it can handle the dynamic and complex characteristics of the client. Factors such as the distribution of client data, computing power, and training performance can all influence  $\alpha$ . The quantity-aware branch integrates these complex characteristics into the client’s embedding vector, allowing for a more accurate prediction of  $\alpha$  and making the model more adaptable to the diversity of different clients. Second, it improves the robustness of the model. Direct calculation of  $\alpha$  can lead to large deviations when there is noise or an attack during model training, which affects the aggregation quality of the global model. By learning the mapping from the embedding vector to  $\alpha$ , the model can tolerate noise or abnormal data, enhancing the system’s robustness and fault tolerance. Third, the proposed method is highly scalable. The quantity-aware branch, as a module, can seamlessly integrate with other components of the model.

$$\alpha = \frac{\Delta\theta_i}{\eta \cdot \mathbb{E}[\nabla L(\theta_i)] \cdot |D_i|} \quad (12)$$

### B. Reliability of the Adjustment Factor

With its powerful computing capability, the server can calculate the trend of the adjustment factor  $\alpha$  predicted by the quantity-aware branch under different training data volumes. Through experiments, we found that, under the same batch size, learning rate, and loss function, the  $\alpha$  value of each communication round of the training model varies with different data volumes, as shown in Fig. 3. From Fig. 3, we observe significant differences in the size of the  $\alpha$  value across communication rounds, but the distribution of the  $\alpha$  value remains similar. Therefore, the server can acquire prior

knowledge of the  $\alpha$  value distribution through pre-training and use this information to verify whether the data volume reported by the client is accurate. If the reported data is found to be false, the server can issue an early warning to the client to discourage dishonest behavior.

Specifically, the server can predict the distribution of the predicted  $\alpha$  values under different data volumes by fitting or using convolutional neural networks. The authenticity of the  $\alpha$  value uploaded by the client can then be assessed based on the predicted distribution.

## V. EXPERIMENTAL EVALUATION

### A. Experiment Setup

In the experiment, we set the number of clients to 100 and selected 10 clients in each round to participate in federated training. One client was set to report three times the amount of data to the server in each training round, simulating a scenario with a dishonest client. Other experimental settings are as follows.

**Datasets.** We selected CIFAR-10 [23] and the medical dataset MedMNIST for the experiments. CIFAR-10 is an image dataset with 10 categories, containing 50,000 training samples and 10,000 test samples, all of which are 3-channel images of size 32\*32. MedMNIST [24] is a medical dataset that includes 12 2D sub-datasets and 6 3D sub-datasets, with image types covering X-rays, CT scans, MRIs, and more. For this experiment, we selected two 2D sub-datasets from MedMNIST: OrganaMNIST [25] and PathMNIST [26].

**Comparison algorithms.** In this paper, we selected four classic FL algorithms: FedAvg, FedProx, Ditto, and Scaffold. Among them, FedAvg is the most fundamental FL algorithm and introduced the concept of FL. The FedProx algorithm adds a regularization term to FedAvg to address the heterogeneity problem in FL. The Ditto algorithm deploys both personalized and global models on the client, using the global model as a constraint on the personalized model during training to prevent the personalized model from deviating too far from the global model. The Scaffold algorithm introduces control variables and

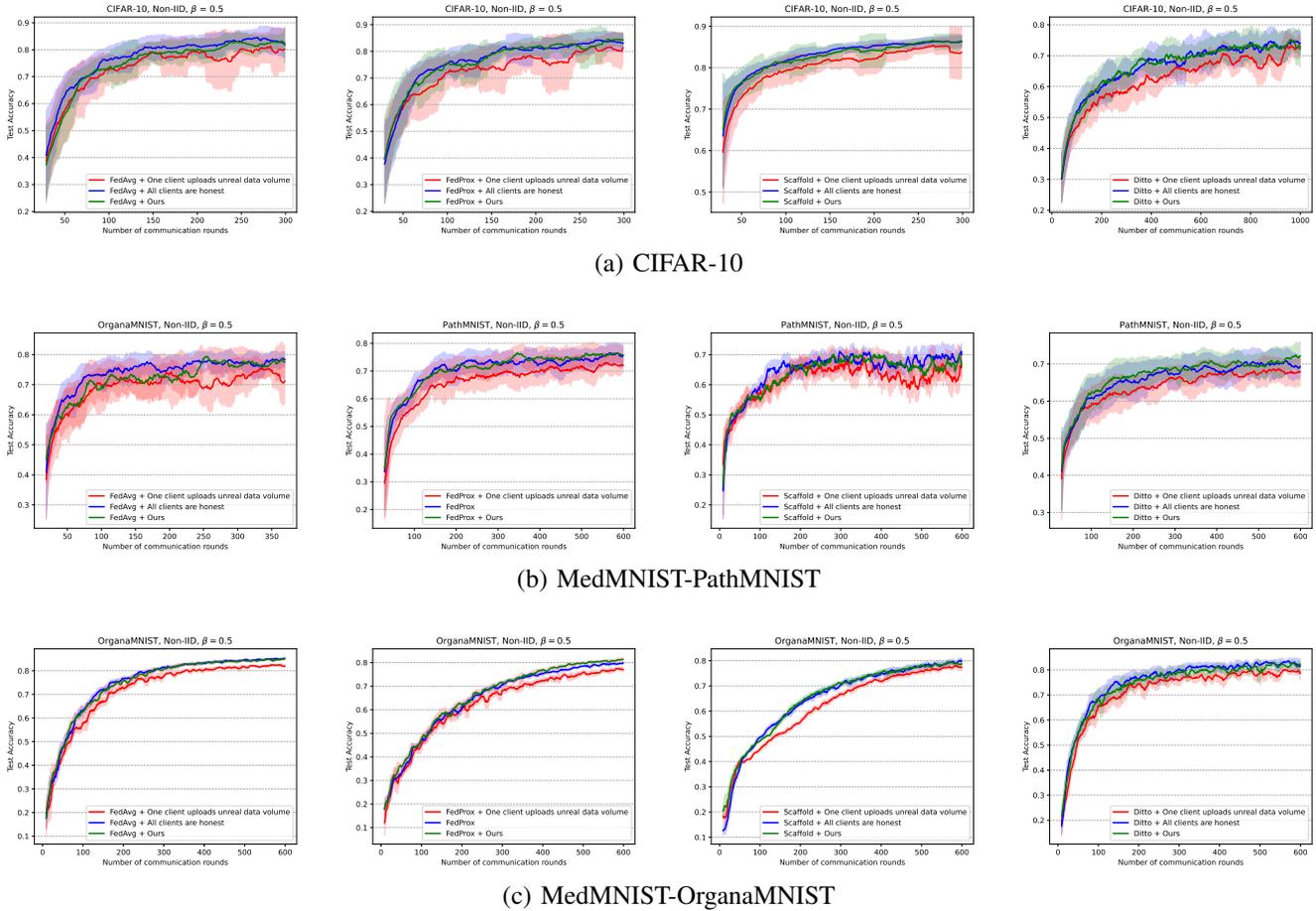


Fig. 4. The proposed algorithm is tested with four FL methods: FedAvg, FedProx, Scaffold, and Ditto. Two scenarios are considered: (1) all clients behave honestly, and (2) one client uploads falsified data volumes. The evaluation measures test accuracy on the CIFAR-10, MedMNIST-PathMNIST, and MedMNIST-OrganMNIST datasets. The datasets are non-IID, with a Dirichlet distribution ( $\beta = 0.5$ ) used to define the data distribution.

control gradients, ensuring model stability even in scenarios with heterogeneous data.

### B. Results and Discussion

We evaluate four FL algorithms—FedAvg [7], FedProx [13], Scaffold [15], and Ditto [14]—on the three datasets mentioned above to compare the accuracy of the proposed algorithm under two scenarios: when all clients are honest and when one client uploads unreal data volume. In the experiment, we use a Dirichlet distribution to partition the Non-IID data for each client, with  $\beta = 0.5$ . The experimental results are shown in Fig. 4.

From Fig. 4, we can observe that when a client uploads an unreal amount of training data during FL, it negatively affects the performance of the aggregated global model. For example, in the Cifar-10 dataset, when a client reports an unreal amount of training data, the accuracy of the global model after aggregation decreases by 2.58%, 3.79%, 3.2%, and 3.51% for the four FL algorithms—FedAvg, FedProx, Scaffold, and Ditto—compared to the weighted aggregation based on the client’s actual data volume. Furthermore, we can see that

when the client uploads unreal data, the accuracy of the global model fluctuates significantly during training. This is because unreal reporting of client data volume leads to inaccurate weighted aggregation, which over-amplifies the influence of dishonest clients. In Non-IID settings, clients that report unreal data volumes may provide data that differs significantly from the global distribution, causing the global model to deviate from the true characteristics of the overall data, thus reducing accuracy. During training, the model updates become unstable due to the unreal data volume, and the imbalance in weights leads to frequent adjustments in the global model, resulting in large accuracy fluctuations.

Similarly, when clients dishonestly report the amount of training data, the accuracy of the global model using the proposed method, FedDua, remains nearly the same as when the data is honestly reported. This is because, in the FedDua algorithm, the server can verify the authenticity of the training data uploaded by each client based on the distribution of the adjustment factor  $\alpha$ , combined with the prior knowledge of  $\alpha$  distribution obtained through pre-training. If a dishonest client is detected, the server will issue a warning. If the client

continues to upload false data volume, it will be excluded from federated training.

### Computation and communication complexity analysis.

The proposed FedDua algorithm introduces two additional computations: client training of the data quantity-aware branch and server pre-training of prior knowledge about the  $\alpha$  distribution. The server possesses sufficient computing, storage, and communication capabilities. Therefore, the additional computational cost primarily arises from client training of the data quantity-aware branch. Typically, the number of parameters in the data quantity-aware branch model,  $\varphi$ , is much smaller than the local model  $\theta$  on the client, and the corresponding computational complexity is  $O(\frac{1}{d}\theta)$ , with  $d > 10$ . As a result, the additional computational cost for the client due to the data quantity-aware branch is less than 10%. Furthermore, no additional communication cost is incurred.

## VI. CONCLUSIONS

In the proposed FedDua algorithm, the data quantity-aware branch and server pre-training enable accurate judgment of the authenticity of the data uploaded by the client and prediction of the client's data volume. The proposed method addresses the issue of performance degradation in the global model due to clients dishonestly reporting their data volume during weighted aggregation in FL. However, this paper focuses only on dishonestly reported data volume. The quality of the data also significantly impacts the performance of the aggregated global model. In future work, we will analyze weighted aggregation in FL from the perspective of data quality to further improve model performance.

## VII. ACKNOWLEDGMENTS

This work is funded in part by an international Collaboration Fund for Creative Research of National Science Foundation of China (NSFC ICFCRT) under the Grant no. W2441019; National Natural Science Foundation of China (62136003); Shanghai Pujiang Program (22PJ1423400) and Shanghai Sailing Program (22YF1401300); the Ministry of Education, Singapore, under its Academic Research Fund Tier 1; the National Research Foundation, Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No. AISG2-RP-2020-019).

## REFERENCES

- [1] Murray Shanahan, Kyle McDonell, and Laria Reynolds, "Role play with large language models," *Nature*, vol. 623, no. 7987, pp. 493–498, 2023.
- [2] Rajani Singh, Ashutosh Dhar Dwivedi, Gautam Srivastava, Pushpita Chatterjee, and Jerry Chun-Wei Lin, "A privacy-preserving internet of things smart healthcare financial system," *IEEE Internet of Things Journal*, vol. 10, no. 21, pp. 18452–18460, 2023.
- [3] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li, "End-to-end autonomous driving: Challenges and frontiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [4] Randy Goebel, Han Yu, Boi Faltings, Lixin Fan, and Zehui Xiong, Eds., *Trustworthy Federated Learning*, Springer, Cham, 2023.
- [5] Yaochu Jin, Hangyu Zhu, Jinjin Xu, and Yang Chen, *Federated Learning*, Springer, 2023.
- [6] Tao et al. Fan, "Ten challenging problems in federated foundation models," *IEEE Transactions on Knowledge and Data Engineering*, 2025.
- [7] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.
- [8] Leming Wu, Yaochu Jin, Yuping Yan, and Kuangrong Hao, "Fl-otcsenc: Towards secure federated learning with deep compressed sensing," *Knowledge-Based Systems*, vol. 291, pp. 111534, 2024.
- [9] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán, "Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges," *IEEE Communications Surveys & Tutorials*, 2023.
- [10] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu, *Federated Learning*, Springer, Cham, 2020.
- [11] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S. Yu, "Privacy and robustness in federated learning: Attacks and defenses," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 35, no. 7, pp. 8726–8746, 2024.
- [12] Chao Ren, Han Yu, Hongyi Peng, Xiaoli Tang, Bo Zhao, Liping Yi, Alysa Ziyang Tan, Yulan Gao, Anran Li, Xiaoxiao Li, et al., "Advances and open challenges in federated foundation models," *IEEE Communications Surveys and Tutorials*, 2025.
- [13] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.
- [14] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith, "Ditto: Fair and robust federated learning through personalization," in *International conference on machine learning*. PMLR, 2021, pp. 6357–6368.
- [15] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*. PMLR, 2020, pp. 5132–5143.
- [16] Lingjuan Lyu, Xinyi Xu, Qian Wang, and Han Yu, "Collaborative fairness in federated learning," *Federated Learning: Privacy and Incentive*, pp. 189–204, 2020.
- [17] Yiqiang Chen, Xiaodong Yang, Xin Qin, Han Yu, Piu Chan, and Zhiqi Shen, "Dealing with label quality disparity in federated learning," *Federated Learning: Privacy and Incentive*, pp. 108–121, 2020.
- [18] Guan Wang, Charlie Xiaoqian Dang, and Ziyi Zhou, "Measure contribution of participants in federated learning," in *2019 IEEE international conference on big data (Big Data)*. IEEE, 2019, pp. 2597–2604.
- [19] Yuxin Shi, Zelei Liu, Zhuang Shi, and Han Yu, "Fairness-aware client selection for federated learning," in *2023 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2023, pp. 324–329.
- [20] Liping Yi, Wang Gang, and Liu Xiaoguang, "Qsfl: A two-level uplink communication optimization framework for federated learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 25501–25513.
- [21] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.
- [22] Zexi Li, Tao Lin, Xinyi Shang, and Chao Wu, "Revisiting weighted aggregation in federated learning with neural networks," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19767–19788.
- [23] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," 2009.
- [24] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni, "Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification," *Scientific Data*, vol. 10, no. 1, pp. 41, 2023.
- [25] Patrick Bilic, Patrick Christ, Hongwei Bran Li, Eugene Vorontsov, Avi Ben-Cohen, Georgios Kaissis, Adi Szeskin, Colin Jacobs, Gabriel Efrain Humpire Mamani, Gabriel Chartrand, et al., "The liver tumor segmentation benchmark (lits)," *Medical Image Analysis*, vol. 84, pp. 102680, 2023.
- [26] Jakob Nikolas Kather, Johannes Krisam, Pornpimol Charoentong, Tom Luedde, Esther Herpel, Cleo-Aron Weis, Timo Gaiser, Alexander Marx, Nektarios A Valous, Dyke Ferber, et al., "Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study," *PLoS medicine*, vol. 16, no. 1, pp. e1002730, 2019.