Apply SplitFed to FedCorr: Detecting Unreliable Edge Learning Users in 6G Network Systems

Geng Tian
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
12332463@mail.sustech.edu.cn

***Abstract*—In the two surveys [1] [2] I have chosen, a major issue arises: how to address the problem of unreliable users in edge learning within 6G network systems. A plethora of unreliable users participate in edge computing, and their utilization of faulty data to train models poses severe threats to the global model's performance. Thus, detecting these unreliable users is imperative.**

The algorithm I utilized for comparison, was presented at the CVPR conference in 2022 [3], which is named FedCorr. FedCorr furnishes a noise data detection approach for users. This method enables the identification of users in edge learning who utilize noisy data, referred to as noisy data users. The technique can amend the users' noisy data to regular data and permit training of the model using corrected data, guaranteeing adequate data quantities necessary for model training. Whilst completing this series of operations, FedCorr does not directly request users' data and therefore does not violate users' privacy. Two bottlenecks exist in FedCorr: firstly, non-IID data cannot be recognized; secondly, fraud by unreliable users cannot be recognized. This paper proposes two methods, replacing FL with SFL and binding data detection with model training, to tackle the issues faced by FedCorr.

***Index Terms*—SplitFed, 6G, privacy, unreliable users, noisy data, non-IID data, edge intelligence, edge computing**

I. Background

In the practical implementation of 6G communications, edge computing plays a crucial role in the network system [1]. Centralized computing servers are no longer appropriate for 6G networks as network traffic continues to increase. In contrast, edge computing, which emphasizes real-time response and throughput, is becoming the mainstream computing method for 6G networks.

Nonetheless, security poses a significant challenge in all 6G networking systems. Edge computing is susceptible to cyberattacks. Data can be contaminated by attackers, rendering many users involved in edge computing unreliable. Since edge computing systems can ensure user privacy, identifying issues with user data across the entire system is complicated. Unreliable users generate data of low quality during training, and failing to detect these inaccuracies can adversely affect the final model by committing it with inaccurate or invalid data, causing a reduction in accuracy. Unreliable user-provided data can be divided into two categories: non-IID data [4] and noise data [5].

Non-IID data refers to data that does not conform to the independent and identically distributed (IID) properties between different users. To achieve better training results in edge computing, it is essential that users provide consistent data for model training. However, untrustworthy users may utilize data that is distributed differently to other users, potentially impacting the stability and precision of the model. Additionally, accurate labeling within the training data for AI models is necessary to guarantee that appropriate features are acquired. If an attacker intentionally alters the labels within the user's training data to convey erroneous information, it can significantly impact the training outcomes of the model. To identify unreliable users among many, it is crucial to establish whether anomalies exist in each user's data accurately.

SplitFed(SFL) is an edge learning framework that combines federated and partition learning [6]. It follows the tradition of not sharing data between users in edge learning, which protects users' privacy. However, this practice also creates obstacles when identifying untrustworthy users. SplitFed addresses this issue by enabling the server to access intermediate results of training and data labels, thereby providing more opportunities for the server to check the quality of user data.

The primary advantage of edge computing lies in safeguarding user privacy, barring the server from acquiring any user data, and assessing the quality of the user data directly. During SplitFed training, the server receives the user's forward propagation outcome, which may lead to data inference by the server and augment the possibility of detecting disreputable users.

The algorithm I utilized for comparison, presented at the CVPR conference in 2022 [3], developed a Federated Learning-based Edge Computing framework known as Fedcorr. This framework identifies and rectifies user noise data to assist in the screening process for users with abnormal data in 6G network systems.

II. Analysis of FedCorr

A. Introduction of FedCorr

FedCorr is an edge learning framework based on federated learning. Its purpose is to identify the presence of noisy data among users, allowing potentially unreliable users to be identified. Simultaneously, it can rectify incorrect labels in disordered data, enabling the system to utilize undependable user data for model training. These steps ensure that the edge learning system can detect and correct noisy data, preventing the impact of unreliable users on the global model. The model's training framework consists of three stages, with a specific process as outlined below.

In the first phase, the goal is to select untrustworthy users. FedCorr has the ability to detect clients containing noisy data and predict the subspace dimensions from the user's independently measured model, thus establishing the presence of noisy data. In this procedure, every user must utilize their local information to train the model. The model, after training, generates predictions on the test set, which are sent to the server. Once the prediction results have been obtained, the server calculates the dimension of each client's model prediction subspace utilizing LID [7]. With this calculation, FedCorr identifies users with noisy data.

LID (Local Intrinsic Dimension) is a technique for evaluating the local dimension of a data manifold. In simple terms, LID can help us estimate the count point's local dimension, aiding in our understanding of the data's internal structure. The LID calculation relies on the nearest neighbor distance of the data point, and it estimates the local dimension by assessing the distance ratio between the data point and its closest neighbor. In the FedCorr algorithm, LID is applied to assess the quality of the dataset for each client and to dynamically detect clients that introduce noise. This approach enables FedCorr to effectively rectify label noise [8] [9].

In the second phase, the objective was to evaluate the degree of inaccuracies in the labels of the untrustworthy user. FedCorr implements an adaptable local regularization term established on evaluated local noise levels to process data with distorted labels and strengthen the model training's stability. This regularization term is determined by using the fraction of LID, which was derived from stage one. Specifically, FedCorr calculates the local noise level for each customer and adjusts the regularization terms of the local model accordingly. This adaptive local regularization term reduces the influence of label noise on model training and enhances model accuracy and robustness by correcting false labels in user data.

In the third phase, FedCorr fine-tunes certified and reliable users. FedCorr fine-tunes the global model using identified noise-free data to enhance its performance. The model is then trained using raw data from trustworthy users and corrected data from untrustworthy users. The noise data is retained after label correction to train the model instead of eliminating the users with noisy data. This approach aims to utilize the full potential of the user's data to enhance model reliability.

B. Advantages of FedCorr

Firstly, FedCorr has the ability to detect noisy data that is present locally to users while safeguarding their privacy. Consequently, it can identify untrustworthy users engaged in edge computing.

Secondly, FedCorr utilized LID to estimate the level of label noise within the data provided by each unreliable user. Subsequently, FedCorr corrected the labels of any noisy data to adhere to standards. This enabled unreliable users to continue participating in edge computing, maximize client data usage, and increase the model's overall data volume. FedCorr opts to correct noisy data from unreliable users rather than discarding it, ensuring that edge computing can still operate even if an attacker breaches a large-scale user. If the edge learning framework disregards data from all unreliable users, then the model cannot be adequately trained when too many unreliable users exist. Failure of the model to train appropriately is the worst-case scenario, and it is more perilous than a decrease in model accuracy. In general, FedCorr can detect and correct noisy data that exists locally to unreliable users, without violating the user's personal privacy in the process.

C. Bottleneck of FedCorr

Firstly, it must be noted that FedCorr is limited in its ability to detect unreliable users with non-IID data, only capable of detecting those with noisy data. This limitation leads to a significant number of unreliable users remaining undetected. FedCorr utilizes LID to compute scores, which serves to distinguish noisy data but cannot identify the existence of non-IID data. Generally speaking, non-independent and identically distributed (non-IID) data can be divided into two categories. The first is uneven data labeling, where some data labels may be significantly more or less than others. The second is unequal data volume, where some users may have significantly more or less data than others. These unreliable non-IID data users that evade detection will be considered reliable users by the system to participate in edge computing, which has a significant impact on the final generated model [10]. At present, mainstream edge learning frameworks have the ability to deal with non-IID data, but first the system should be able to distinguish which user data is non-IID.

From the earlier discussion, it is apparent that undependable users hinder the model training process in edge computing primarily by supplying untrustworthy data, which comprises mainly non-IID and noisy data. In real-world scenarios, non-IID data occurs far more frequently than noisy data. Therefore, an edge learning structure to detect non-IID data is devised. In the best case, such a framework should be able to recognize both non-IID data and data that contains noise, so that hidden unreliable users can be found as much as possible.

The operation of FedCorr necessitates user cooperation, rendering it more challenging to identify undependable users, and creating additional openings for attackers. During the primary stage of FedCorr, each user uploads personal outcomes of a locally trained model, which the server utilizes to evaluate whether the user holds erroneous data. Good users will follow the server's instructions to upload model test results. Attackers posing as users, however, will upload fake results in an attempt to deceive the server. The server is therefore unable to distinguish between reliable and unreliable users.

To address these issues, a logic must be designed to prevent users from submitting fraudulent data, which can help mitigate the exploitation of vulnerabilities in edge computing frameworks. If a perpetrator can impersonate a trustworthy user, they will engage in more aggressive measures to taint data, without concern for anomalous data being promptly detected by the system.

III. Some Improvements to the Original Method

A. Use SplitFed Learning Instead of Federated Learning

This section examines the ability of edge learning frameworks to identify untrustworthy users with non-iid data.

SplitFed learning (SFL) is currently the most popular edge computing algorithm, widely used in everyday life, and offers greater user security and performance than traditional federation learning (FL).

SFL combines traditional federated learning with segmented learning. By absorbing the benefits of both federated learning(FL) and split learning(SL), SFL creates a set of training methods that consider training speed and safety equally. Typically, only edge users participate in FL training, and FL model training does not involve a central server. The server's task in FL is to aggregate the local models of diverse edge users, generate a unified model, and dispense these models to edge users. In this context, all user data merely necessitates local training, eliminating the need for data exchange with other devices, and communication between the user and the server is exclusively for uploading the trained local model and downloading the consolidated global model. In SFL, the server participates in training along with the user. Unlike traditional machine learning, servers in SFL do not have access to the data and labels. In SL, the neural network will be divided into two parts: half A and half B. Half A will be located on the user's local system, while half B will be positioned on the server. Initially, the user will utilize the data to calculate A locally. After that, the intermediate result of the calculation and the data label will be conveyed to the server. Subsequently, the server will utilize the acquired results to calculate B. The unified backpropagation and update of the neural network can only occur after both parts have been computed. Then the user will transfer their local A to the next user requiring training, who will subsequently carry out the aforementioned operation. In SL, parallel computing is not possible as the system only permits one user to be saved at any given time [11].

SFL combines FL and SL to allow a central server to assist users in model training, while also enabling parallel computing among users. The SplitFed learning training process comprises three distinct stages. Initially, the user conducts forward propagation on the local model using their own data before transmitting the calculation outcomes to the server. The server utilizes the outcomes obtained from the user's computations and labels to propagate forward and backward within the server model. Subsequently, these results are backpropagated to the user. In the end, the user employs the server's backpropagation outcomes and the local data label to backpropagate and update their local model. Afterward, all users' local models are combined to guarantee uniform model distribution.

A crucial technology inherited by SFL from SL is the transmission of tags by users to the server during training. It's worth stressing that the user's data label uploaded to the server doesn't breach the edge computing principle that safeguards user privacy. This is because the server can solely deduce the type of data the user possesses from the label, without knowing the actual data.

When identifying untrustworthy users, uploaded tags can be used to detect whether the data provided by the customer constitutes non-IID data. Key detection indicators include the number and type of data labels. For instance, if a user has a small quantity of data labels, it can be inferred that their data is non-IID data. If the user persists in employing the original data to take part in the model training, the subpar data produced by the user will jeopardize the stability of the global model. Another potential example is when a user's data tag comprises excessive labels of a particular type, implying that the user's data deviates from uniformity and can be categorized as non-IID data. If the user persists in participating in model training utilizing the initial data, the model's generalization capacity will be reduced.

From the above analysis, it is evident that utilizing SFL as opposed to FL as the fundamental framework of edge learning can successfully identify users' local non-IID data, consequently intercepting a greater number of unreliable users.

B. Bind the Detection of User Data to Model Training

This section aims to prevent users from unreliable sources cheating the server to avoid detection.

In FedCorr, there is a significant drawback where users must participate in detecting the quality of their data. They are required to locally pre-train the model using their data and then test it before uploading the results to the server. There is a significant issue in this process. When identifying untrustworthy users, subjective evaluations must be avoided unless explicitly acknowledged as such. If not, unreliable users may appear to cooperate with the system for detection while still being disguised by attackers.

In the new edge learning methodology, the SFL algorithm is implemented rather than FL. To complete the training, cooperation from the server is required by SFL. This enables the identification of data containing noise by detecting the training data transmitted to the server, thus preventing untrustworthy users from deceiving the system. Here is a practical example that demonstrates how this new detection method can deter fraud committed by untrustworthy users for better understanding. Suppose a malicious user, named A. User A has access to regular data that can be employed in edge learning. In order to interfere with the system, User A intentionally alters some of the labels associated with this normal data, thereby creating noisy data. During data transmission to the server, User A opts to use uncontaminated data for model training to evade detection. Subsequently, the results are sent to the server. At this stage, it successfully deceives the server, which makes it difficult for the system to identify it as an unreliable user. However, by the time it is prepared to assault the model with disruptive data, it is already too late. The server will then train on the normal data calculation results that were uploaded instead of requesting the user to submit the calculation results applied for training again. If they use noisy data to train the model, the system will identify it. Conversely, using normal data to train the model will fail to achieve the objective of an attack. However, their own data proves useful in training the global model. Unreliable users face a dilemma at this stage. By adopting this approach, fraud by unreliable users can be effectively prevented.

The selected approach by FedCorr involves computing LID based on the anticipated outcomes of the user's model that has been trained. Here, we aim to map and determine LID for the results of forward propagation that have been sent by the user to the server, with the purpose of ascertaining the presence of any corrupted data.

In practical scenarios of edge computing, there exists a significant proportion of untrustworthy users who indulge in fraudulent activities. By integrating data quality detection and model training, the risk of fraudulent users causing harm to the overall edge computing training can be effectively mitigated.

IV. Conclusion

FedCorr presents an edge learning framework founded on Federated Learning (FL) that enables the identification and rectification of noisy data in unreliable users, while also training the model with the corrected data. However, FedCorr has two limitations related to detecting data from unreliable users: it is incapable of detecting non-IID data and is unable to prevent fraud committed by unreliable users. The existence of these two limitations makes FedCorr not well adapted to complex scenarios and provides many obstacles to its deployment in real network systems. In optimizing the edge learning framework, the use of SFL instead of FL enables the system to detect non-iid data present in the user. By binding the quality of detection data to the model's training process, fraudulent behavior in the detection process can be avoided. These two measures considerably enhance the sensitivity and robustness of FedCorr.

References

[1] H. Guo, J. Li, J. Liu, N. Tian, and N. Kato, "A survey on space-air-ground-sea integrated network security in 6g," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 53–87, 2021.

[2] B. Mao, J. Liu, Y. Wu, and N. Kato, "Security and privacy on 6g network edge: A survey," *IEEE Communications Surveys & Tutorials*, 2023.

[3] J. Xu, Z. Chen, T. Q. Quek, and K. F. E. Chong, "Fedcorr: Multi-stage federated learning for label noise correction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10184–10193, 2022.

[4] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[5] X. Fang and M. Ye, "Robust federated learning with noisy and heterogeneous clients," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10072–10081, 2022.

[6] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 8485–8493, 2022.

[7] M. E. Houle, "Dimensionality, discriminability, density and distance distributions," in *2013 IEEE 13th International Conference on Data Mining Workshops*, pp. 468–473, IEEE, 2013.

[8] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," *arXiv preprint arXiv:1801.02613*, 2018.

[9] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. Erfani, S. Xia, S. Wijewickrema, and J. Bailey, "Dimensionality-driven learning with noisy labels," in *International Conference on Machine Learning*, pp. 3355–3364, PMLR, 2018.

[10] 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.

[11] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtepe, H. Kim, and S. Nepal, "End-to-end evaluation of federated learning and split learning for internet of things," *arXiv preprint arXiv:2003.13376*, 2020.