Comparison of Federated Learning Global Model Aggregation Algorithms

FU SHI-TONG
National Cheng Kung University

LAI GUANG-YU
National Cheng Kung University

Abstract

This study investigates the impact of different aggregation algorithms on Federated Learning (FL) in scenarios involving non-IID datasets. FL enables collaborative model training while preserving data privacy, making it particularly relevant for sectors like healthcare. However, the non-IID nature of client data can lead to challenges such as overfitting and reduced model accuracy. We compare four aggregation algorithms—FedAvg, ABAVG, ACC_inverse, and IDA using non-IID datasets with varying label distributions. The MNIST dataset is used for testing, and both MLP and CNN models are trained. Results show that ABAVG achieves the highest accuracy (91.02%) in the MLP model, while ACC inverse, though having the lowest loss, demonstrates overfitting with lower accuracy. In the CNN model, Salmeron et al. *method yields the best accuracy (97.04%), despite higher loss. These findings highlight the importance of choosing appropriate aggregation strategies to optimize model performance in non-IID environments.*

1. Introduction

Federated Learning (FL)[1], as a decentralized machine learning technology, allows multiple nodes to collaboratively train a model without sharing data. Each participant only sends locally computed model updates to a central server for aggregation, without transmitting raw data. This technology is particularly suited for scenarios where data privacy needs to be protected, such as in the financial and healthcare sectors. The decentralized structure of FL avoids the risks associated with centralized data storage, significantly reducing the likelihood of data breaches. According to relevant literature [2], the potential of FL in medical applications has gained widespread attention, especially in terms of improving diagnostic model accuracy while preserving patient privacy.

In the training process of Federated Learning, the workflow is divided into a client part and a server part. Taking the training of a diagnostic model as an example, the clients represent the participating hospitals, each having its own set of cases. Data is not shared between clients (hospitals). Each client independently uses its own training data (cases) to train model parameters, which are then encrypted and sent to the server. The server decrypts the received parameters and employs an aggregation algorithm to combine the parameters from all clients into a global parameter. This global parameter is then distributed back to all clients to update their local models. This process is repeated iteratively.

The above process can be divided into the following steps:

1. **Local Model Training**: Each participating entity uses its local data to train a deep learning model, ensuring that sensitive data remains within the organization. In a Federated Learning setup, the local models, which could utilize architectures like MLP for certain tasks or CNN for image-related tasks, are trained independently. These locally trained models then share their updates with the central server for aggregation, without compromising the privacy of the original data.
2. **Parameter Updates**: After training, each institution encrypts and sends the parameters of its local model (such as weights and biases) to the central server.
3. **Global Model Aggregation**: The central server receives the encrypted parameters from each participant and applies the aggregation algorithm to perform a weighted average, generating a global model update. This process ensures that the privacy of each participant's data is preserved, as only model updates are shared, not the raw data. The updated model parameters are then returned to the participants, allowing them to continue local training with the latest global model. This collaborative learning process not only improves the model's accuracy but also ensures the security and privacy of the data, as it never leaves the local environment.

However, in Federated Learning (FL), the issue of non-IID datasets often arises. Non-IID refers to datasets where the random variables do not satisfy the criteria of being independent or identically distributed. This concept is crucial in statistics and machine learning, as it can significantly affect model performance and training strategies. This issue

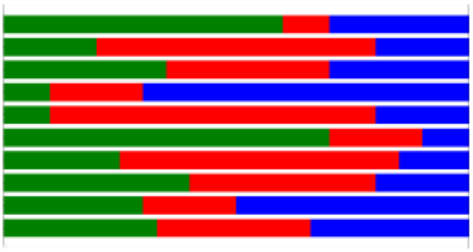


Figure 1. The non-IID situation mentioned in [6] refers to the case where clients have the same labels, but each label contains a different number of samples.

can lead to clients overfitting their own datasets, resulting in a decrease in the accuracy of the aggregated model after parameter merging.

Many studies have proposed methods to address this issue. Some research focuses on enhancing Local Model Training. **FedProx** [3] introduces a regularization term in the loss function to constrain the differences between the local model and the global model. **SCAFFOLD** [4] employs a control variable to correct bias between global weights and local weights. **FedDyn** [5] incorporates the parameters of the global model into the loss function, influencing the weights learned by the local model. These approaches effectively reduce the divergence between local models and the global model, improving overall performance.

In the Global Model Aggregation part, many studies have focused on assigning different weights to clients, allowing each client to have a varying degree of influence on the global model. **ABAVG** [6] assigns a normalized weight to each local model based on its validated accuracy. **Salmeron *et al.*** [7] suggested giving greater weight to local models with lower validated accuracy in order to improve their metrics in their datasets. **IDA** [8] uses the inverse of the distance between local parameters and the averaged parameters as weights. This approach allows the model to reject or assign less weight to poisoning models, i.e., out-of-distribution models.

In this study, we focus on and compare different aggregation algorithms. Specifically, we examine the differences between various aggregation algorithms under non-iid datasets. The goal is to enhance the global aggregation process in federated learning.

2. Method

In this section, we will first discuss the algorithms we aim to compare. We will utilize four different algorithms, with the details outlined in Sec. 2.1. Next, we will train each algorithm using the same dataset and parameters. The datasets will have the same number of samples but different label distributions, ensuring that the datasets are non-iid. Finally, we will compare the differences in accuracy and loss among these algorithms.

2.1. Aggregation Algorithm

FedAvg [9] is the first aggregation algorithm used in federated learning. It assigns a normalized weight to each client based on the size of its dataset, which is applied to the local weight. However, due to the non-iid nature of the data, it may lead to overfitting issues in the local models.

$$\omega_g^t = \sum_{i=1}^{n} \frac{d_i}{\sum_{k=1}^{n} d_k} \omega_i^{t-1} \tag{1}$$

ω_g^t denotes the weights of global model in epoch t. n denotes the number of clients. d_i denotes the number of data points on client i. ω_k^i denotes the weights of i^{th} client's model in epoch t.

ABAVG [6] assigns normalized weights to local models based on their validated accuracy. This approach allows local models with higher accuracy to have a greater influence on the global model.

$$\omega_g^t = \sum_{i=1}^{n} \frac{acc_i}{\sum_{k=1}^{n} acc_k} \omega_i^{t-1} \tag{2}$$

acc_i denotes the validated accuracy of i^{th} client's model.

Salmeron *et al.* [7] used the normalized inverse accuracy (**ACC_inverse**) as the weighting coefficient, attempting to assign higher weights to local models with lower accuracy in order to improve their metrics on their respective datasets.

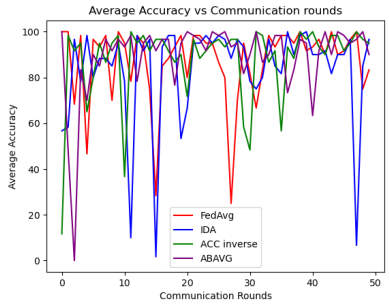
$$\omega_g^t = \sum_{i=1}^{n} \frac{1/acc_i}{\sum_{k=1}^{n} 1/acc_k} \omega_i^{t-1} \tag{3}$$

IDA [8] uses the n the inverse distance of each client parameters to the average model of all clients. This approach allows the model to reject or assign less weight to the out-of-distribution models.

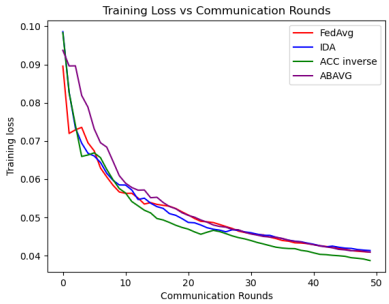
$$\omega_g^t = \sum_{i=1}^{n} \frac{1}{Z} ||\omega_{avg}^{t-1} - \omega_i^{t-1}||^{-1} \omega_i^{t-1} \tag{4}$$

$$Z = \sum_{k=1}^{n} ||\omega_{avg}^{t-1} - \omega_k^{t-1}||^{-1} \tag{5}$$

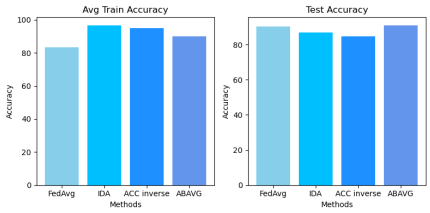
We compared the above methods by analyzing the changes in accuracy and loss after multiple interactions between the clients and the server. Additionally, we will evaluate the variations across different models to validate the observed results.



(a) Training Accuracy

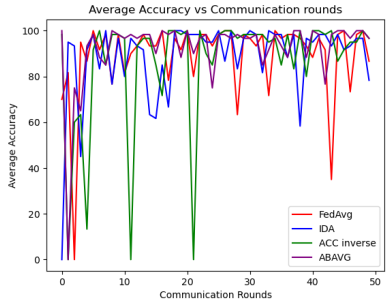


(b) Training Loss

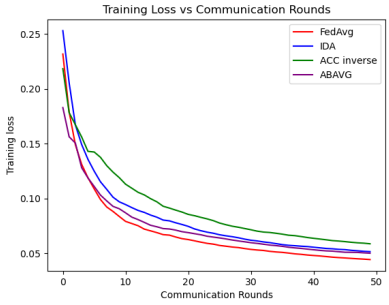


(c) Average Training Accuracy and Test Accuracy

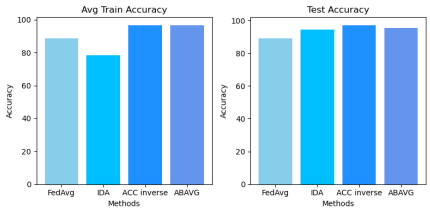
Figure 2. MLP result. (a) presents the training accuracy of four different methods. The graph illustrates how each method's accuracy evolves over time during training. (b) displays the training loss across four different methods. It shows how the loss decreases over time during the training process for each method. By comparing the loss curves, we can evaluate the efficiency of each method in minimizing the loss function and the speed at which they converge



(a) Training Accuracy



(b) Training Loss



(c) Average Training Accuracy and Test Accuracy

Figure 3. CNN result.

3. Experiment

We trained each model using different aggregation algorithms and recorded the changes in accuracy and loss with the number of communication rounds.

Datasets. In this study, we use the MNIST dataset for testing, as it conveniently allows for creating a non-iid effect.

Models. The models used are MLP and CNN. The MLP model uses a single hidden layer with 64 dimensions. The CNN model uses convolutional layers. The first layer has 10 kernels, and the second layer has 20 kernels, with a kernel size of 5 for both layers. Finally, the model is connected to two fully connected layers.

Parameters. The number of epochs is set to 50, which means the clients and server will communicate for 50 rounds. Local batch size is set to 10. We use SGD for the optimizer of models. Learning rate is set to 0.01.

Environment. A single machine is used to simulate the interactions between the clients and the server.

4. Results

4.1. Model : MLP

From the training loss graph Fig. 2b, it can be observed that the loss of three methods (FedAvg, IDA, ABAVG) almost converges to the same value, while ACC inverse's loss converges to a much lower value.

This result is also reflected in the accuracy chart Fig. 2a. The highest accuracy is achieved by ABAVG, with an accuracy of 91.02%. FedAvg is not much lower, with an accuracy of 90.27%. The accuracies of IDA and ACC inverse are 86.80% and 84.75%, respectively.

Combining the results of loss and accuracy, it is clear that the model is overfitting, leading to a decrease in accuracy. ACC inverse, with the lowest loss, presents a perfect example of overfitting, as its lowest accuracy coincides with the lowest loss value.

4.2. Model : CNN

Based on the training loss graph Fig. 3b, it can be seen that FedAvg's loss converges to the lowest value, yet its performance is the worst. Both IDA and ABAVG converge to sim-

ilar values, with higher loss than FedAvg, but their accuracy is better than that of FedAvg Fig. 3a. The best accuracy is achieved by Salmeron et al., despite having the highest loss, which does not affect its performance.

From the results Fig. 3, it can be seen that, unlike MLP, improving the central algorithm in CNN can enhance the model's accuracy. Among them, ACC inverse achieves the best accuracy, reaching 97.04%. The accuracies of ABAVG and IDA are 95.29% and 94.46%, respectively, all of which are higher than FedAvg's 89.11%.

5. Conclusion

From the results of both MLP and CNN models, it is clear that improving the central algorithm can lead to better performance, particularly in CNN, where enhanced accuracy is observed with methods like ABAVG and IDA. Although FedAvg achieves the lowest loss in both models, it consistently performs poorly, especially in CNN. On the other hand, ACC_inverse, while exhibiting the lowest loss in the training process, demonstrates overfitting, as evidenced by its significantly lower accuracy despite the minimal loss.

For CNN, ACC_inverse achieves the highest accuracy at 97.04%, surpassing ABAVG and IDA, which achieve accuracies of 95.29% and 94.46%, respectively. These methods outperform FedAvg, which only reaches an accuracy of 89.11%. In contrast, while ACC_inverse shows the lowest loss, this does not necessarily correlate with improved performance, highlighting the issue of overfitting.

In conclusion, optimizing the central algorithm, especially in CNN, leads to improved accuracy, but attention must be paid to overfitting, where a model with the lowest loss may not always result in the best generalization performance.It is crucial to adjust the model and aggregation algorithm according to the specific use case and training objectives in order to avoid issues like overfitting or poor accuracy.

References

[1] J. Konečnỳ, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016. 1

[2] M. Ali, F. Naeem, M. Tariq, and G. Kaddoum, "Federated learning for privacy preservation in smart healthcare systems: A comprehensive survey," *IEEE journal of biomedical and health informatics*, vol. 27, no. 2, pp. 778–789, 2022. 1

[3] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020. 2

[4] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*. PMLR, 2020, pp. 5132–5143. 2

[5] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," *arXiv preprint arXiv:2111.04263*, 2021. 2

[6] J. Xiao, C. Du, Z. Duan, and W. Guo, "A novel server-side aggregation strategy for federated learning in non-iid situations," in *2021 20th international symposium on parallel and distributed computing (ISPDC)*. IEEE, 2021, pp. 17–24. 2

[7] J. L. Salmeron, I. Arévalo, and A. Ruiz-Celma, "Benchmarking federated strategies in peer-to-peer federated learning for biomedical data," *Heliyon*, vol. 9, no. 6, 2023. 2

[8] Y. Yeganeh, A. Farshad, N. Navab, and S. Albarqouni, "Inverse distance aggregation for federated learning with non-iid data," in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning: Second MICCAI Workshop, DART 2020, and First MICCAI Workshop, DCL 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4–8, 2020, Proceedings 2*. Springer, 2020, pp. 150–159. 2

[9] 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