PriMera Scientific Engineering
Volume 3 Issue 6 December 2023
ISSN: 2834-2550

Step-wise Model Aggregation for Securing Federated Learning

Type: Research Article
Received: October 21, 2023
Published: November 23, 2023

Citation:
Shahenda Magdy., et al. "Step-wise Model Aggregation for Securing Federated Learning". PriMera Scientific Engineering 3.6 (2023): 03-09.

Shahenda Magdy*, Mahmoud Bahaa and Alia ElBolock

German International University, Egypt

***Corresponding Author:** Shahenda Magdy, German International University, Egypt.

Abstract

Federated learning (FL) is a distributed machine learning technique that enables remote devices to share their local models without sharing their data. While this system benefits security, it still has many vulnerabilities. In this work, we propose a new aggregation system that mitigates some of these vulnerabilities. Our aggregation framework is based on: Connecting with each client individually, calculating clients' model changes that will affect the global model, and finally preventing aggregation of any client model until the accepted range of distances with other clients is calculated and the range of this client is within it. This approach aims to mitigate against Causative, Byzantine, and Membership Inference attacks. It has achieved an accuracy of over 90 percent for detecting malicious agents and removing them.

Keywords: Federated Learning; Security; Step - wise Model Aggregation

Introduction

In order to make predictions or decisions without being explicitly programmed, Machine learning (ML) algorithms use sample data, or training data, to build a model. These algorithms are used in a wide range of applications, such as medicine, email filtering, speech recognition, agriculture, and computer vision, where it is difficult or impossible to develop conventional algorithms to perform the required tasks. Since clients' data is typically included in exchanged gradients or models, data privacy and confidentiality are a concern in such approaches as these data may hold confidential information.

Federated learning (FL) is a machine learning technique that trains an algorithm across multiple decentralized edge devices or clients holding local data samples, without exchanging them. Training data is partitioned and trained by trainers locally and individually in federated learning. Local gradients are sent to the parameter server which aggregates those gradients and updates global model parameters accordingly. The primary benefit of employing FL approaches to machine learning is the guarantee of data secrecy and privacy. In fact, there is no external upload, concatenation, or exchange of local data. It is more difficult to break into the database because it is divided into local bits externally. To perform that successfully, all parties should ensure that they are dealing with a security system.

Although FL does contain differential-privacy (DP) and many works have been done in this area, it is still vulnerable to many attacks. Many of the works can serve as a powerful first line of defense against partially honest participants who wish to expose the private data of other participants. There is still a huge gap between what is actually needed and the existing privacy-preserving solutions. Participants can now view the intermediate model and provide arbitrary updates as part of the federated learning environment. For example, attackers who impersonate helpful participants may send manipulated updates that maliciously affect the characteristics of the trained model (Causative attack), or a lazy participant may lie about the volume of data records he or she uses for training, causing the algorithm to produce an inaccurate model.

Assuring the integrity of regional training procedures is essential for the effective mitigation of these model-tampered threats on federated learning. In other words, the local training algorithm's output needs to be integrated. There are established defenses to control the participants or explicit observation of the training, such as robust losses and anomaly detection data. Since neither of these presumptions holds true security for federated learning, maintaining integrity is a significant challenge; because each participants' local model parameters are the only ones that the server observes.

To mitigate this, we developed a new integrity system that disallows each client to integrate his local model with the server unless it is completely ensured that they are an honest client. We first let each client send their local model and calculate Euclidean Distance between the client model and the global model. Then we compare all the distances of all the client models and determine the accepted range of distances. Finally, we recheck the distance of each client and according to the result, the system will decide whether to aggregate with the client's model or not.

This is the first step in order to connect with clients. If any maliciousness is detected that client's connection is immediately closed and the client cannot add any updates to the server. The client is removed totally from the system. The approach is expected to mitigate against all Constructive, Byzantine, and Inference attacks.

This paper is structured as follows: Section 2) Related work: which will discuss the previous works that were done to secure the federated learning system. Section 3) Methodology section: explaining our approach. Section 4) Experiment section: A brief explanation of the experiment done, data sets used, and table accurately showing different results of the experiment. Evaluating these results before the conclusion is also in this section. 5) Finally, the conclusion and Future work.

Related Work

Federated learning (FL) enables machine learning models to obtain experience from different data sets located in different sites without sharing training data. This allows personal data to remain on local sites although FL learns from it. Local data samples are not shared. This improves data protection and privacy. The characteristics of the global model are shared with local data centers to integrate the global model into their ML local models. Although, there're still security challenges facing Fl. To improve that, some related works are introduced in the following.

Fang et al. developed a multi-party data protection machine learning framework in [2] that combines homomorphic encryption and federated learning. homomorphic encryption, allows calculations to be performed on encrypted data without decrypting it. The result of the homomorphic operation after decryption is equivalent to the operation on the plain text data.

Another approach is proposed in [5] by Kanagavelu et al. that chooses a subset of FL members as members of the model aggregation committee from the entire member list. Elected committee members then use multi-party computation services to aggregate local models from all FL parties. This provided a mechanism to enable MPC-based model aggregation over the set of model tensors with large parameter sets. Using both Additive and Shamir Secret Sharing MPC protocols. Compared to traditional peer-to-peer frameworks, this is two phases. The MPC-enabled FL framework greatly reduces communication costs and improves system scalability. In the previous 2 works Combining Differential Data Protection with Secure Multiparty Computing/ Homomorphic Encryption allows them to reduce the growth of noise injection by enhancing the number of parties without sacrificing privacy while preserving a predefined confidence level.

Using HE, the proposed scheme in [3] realizes the assessment of data inequality in a way of protecting privacy. Guo et al. propose a secure aggregation protocol that then uses a zero-knowledge proof protocol to offload the task of detecting attacks in a local model from the server to the user. They developed the ZKP protocol that allows users to review models without revealing information about them and without back doors. Their framework thus allows a central server to identify trained model updates without violating the privacy guarantees of secure aggregation.

Brunetta et al. propose a novel approach called NIVA in [1], that enables the distributed aggregation of secret inputs from multiple users by multiple entrusted servers. The result of the aggregation can be verified by the public in a non-interactive manner. In the context of federated learning, for instance, NIVA can be utilized to securely compute the sum of a large number of user data in order to aggregate the model updates for a deep neural network.

A FastSecAgg approach is proposed in [4], which is computationally and communication-ally efficient. FastShare, a novel multi-secret sharing scheme band robust to client dropouts' secure aggregation protocol. FastShare achieves a trade-off between the number of secrets, privacy threshold, and dropout tolerance in order to be information secure. They demonstrate that FastSecAgg is (i) secure against the server colluding with any subset of some constant fraction, such as 10 percent of the clients in the honest-but-curious search, based on FastShare's capabilities; and (ii) is tolerant of a random subset of some constant fraction, such as less than ten percent of clients. While maintaining the same communication cost, FastSecAgg achieves a significant reduction in computation costs. In addition, it ensures security against adaptive adversaries, which can dynamically carry out client corruption during the protocol's execution.

Step-wise Model Aggregation

All the previously presented works show efficient proposed secure aggregation schemes but none of them implemented efficient action against malicious users. Each user's private data is used to train the model, and only local updates are transmitted to the server. In this way, user data is protected while also being used inadvertently to create better models. In our approach, the models sent by each client will be classified as honest client models or malicious client models. Honest clients' models will be classified by 2 filters.

Filter 1: Consistency Filter

Once clients start connecting to the server, they will be added to a stack. The server will start to accept each of these clients' local models individually and in order.

Firstly, after each client sends their model, aggregation happens by converting the server's global model and the client's model to 2 matrices. The matrices will be created from the neural layers shown in Figure??.

Secondly, the system attempts to sum these matrices, divide them by 2 (Getting the average) then convert them to a new model matrix. The system will detect maliciousness if this summation of matrices failed due to different shapes. As shown in Figure?? The purpose of Filter 1 is to detect maliciousness by different shapes of matrices (occurs when using different data-sets to train the models).

Filter 2: Average distances

As shown in Figure 3, after calculating the new model matrix, euclidean distance will be calculated between this new model matrix and the old model's matrix based on paper [6]. The process of this client will be paused, and the whole process will be repeated for all remaining clients. To calculate Euclidean distance of all other clients. Using all these distances, the mean and standard deviation (std) will be calculated. Values within range of the standard deviation added to and subtracted from the mean will be the accepted. Each client model will be checked by comparing its distance with that range. Using this approach the system will be able to detect the change each client added to the global model and claim that extreme changes resulted because of malicious updates of that model.

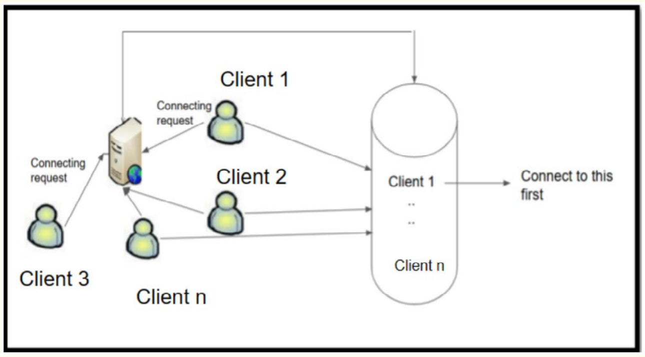

Figure 1: Overview of Connection and Ordering of Clients.

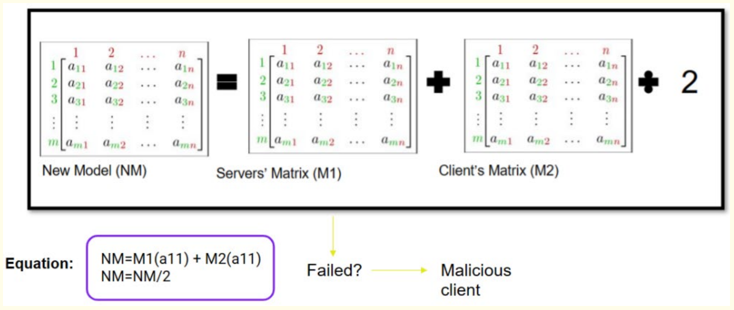

Figure 2: Matrix Summation and Calucaltions.

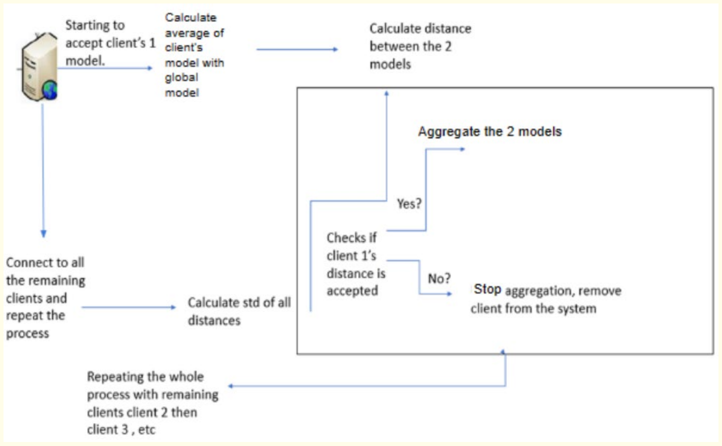
Starting to accept client's 1 model.
Calculate average of client's model with global model
Calculate distance between the 2 models
Aggregate the 2 models
Yes?
Checks if client 1's distance is accepted
No?
Stop aggregation, remove client from the system
Connect to all the remaining clients and repeat the process
Calculate std of all distances
Repeating the whole process with remaining clients client 2 then client 3 , etc

Figure 3: Overview of the whole Filter 2.

After each client's model passes both filters, the aggregation will be done by setting the global model matrix into the new model's matrix.

Experiment

The experiment is done using the MNIST, FMNIST, and CIFAR10 datasets. MNIST is a dataset of handwritten digits that has 60,000 examples and 10,000 examples of handwritten digits from 0 to 9 that have been normalized and centered in a fixed-size image with dimensions (28x28). To train malicious clients that will try to corrupt the actual model, CIFAR-10 Dataset is used. CIFAR-10 is a data-set of images that are used to help computers recognize objects. This dataset will highly affect the MNIST-based model as it contains objects' images and MNIST is based on handwritten digits' images. It includes 60,000 images that are each 32x32 pixels in size.

Fashion MNIST (FMNIST) is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 labeled fashion grey-scale image, associated with a label from 10 classes. Moreover, some random integer arrays were also used to train some of the malicious clients' models.

Experiment Scenario

Different experiments were done, one using 4 clients and the other using 8 clients. Auto - random clients algorithm was done. Different models were implemented and the machine will choose a random model for each of the clients 4. Honest clients models were trained using Keras MNIST library with different distribution for each client. Malicious clients models were trained using Keras Cifar10, FMNIST, and others trained using random inputs arrays.

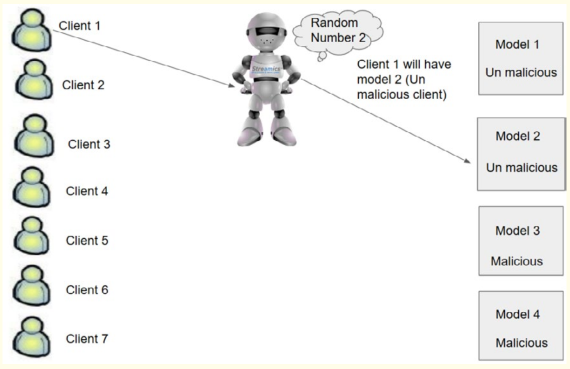

Figure 4: The malicious random clients' machine.

Results

Results show that Filter 1 failed to detect 33% of malicious clients. After applying Filter 2, this rate effectively dropped to 4.7%, however it led to false detection of honest clients as malicious by about 11%; as malicious clients' distances affected calculations of mean and std.

Filter 1 only took 22.5 seconds to aggregate all 8 clients. While using both filters caused it to take an average of 1 min and 2.7 seconds to complete the clients' aggregations. Figure ?? shows the effect of malicious client's enhancement in number on false - negative' rates. Figure ?? shows the effect of malicious clients' enhancement in number on false-positive rates. These show that after using filter 2 false - negative rates are limited and enhancement of malicious clients' numbers can't highly affect it. While false-positive rates are increasing with the enhancement of malicious clients' numbers.

Filter Used	No. of Clients	Detected Clients	Actual Malicious Clients	Time Taken
Filter 1	**4 Clients**	Client 1 Client 3	Client 1 Client 2 Client 3 Client 4	18.4 seconds
	8 Clients	Client 1 Client 2 Client 3 Client 4	Client 1 Client 2 Client 3 Client 4 Client 5 Client 6	20.9 seconds
Filter 1 & 2	**4 Clients**	Client 1 Client 2	Client 1 Client 2	56 seconds
		Client 2	Client 2 Client 3	1 min
	8 Clients	Client 1 Client 2 Client 3 Client 4	Client 1 Client 2 Client 4	43.9 seconds

Table 1: Overview of results.

Conclusion

In this work, we have proposed a very efficient step-wise aggregation system that highly secures federated learning systems. Compared to existing approaches, our system guarantees highly efficient global model updates as it prevents any client from updating the global model until it is ensured that he/she sends appropriate model updates and compares his updates with other clients' updates. And ensure that these updates won't corrupt/ change the global model. Although the excessive presence of malicious models/ clients can affect it, still its effect is very limited and the presence of malicious models/clients in the presence of other honest clients is detected and removed easily. We guarantee that this approach effectively mitigates against Byzantine, Causative, and Membership Inference attacks.

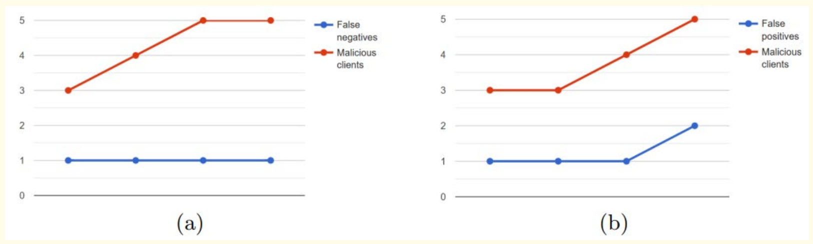

Figure 5: Affect on false negative and false positive rates.

Future work

For future work, we plan to work on more maliciousness presence and try to decrease its effect on false positive results and totally clean its effect on false negative results. Moreover, work on increasing the efficiency of the whole approach to take less time and processes.

References

1. Carlo Brunetta., et al. “Non-interactive, secure verifiable aggregation for decentralized, privacy preserving learning”. In Australasian Conference on Information Security and Privacy, Springer (2021): 510-528.
2. Haokun Fang and Quan Qian. “Privacy preserving machine learning with homomorphic encryption and federated learning”. Future Internet 13.4 (2021): 94.
3. Jiale Guo., et al. “Secure weighted aggregation for federated learning”. arXiv preprint arXiv:2010.08730 (2020).
4. Swanand Kadhe., et al. “Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning”. arXiv preprint arXiv:2009.11248 (2020).
5. Renuga Kanagavelu., et al. “Two-phase multi-party computation enabled privacy-preserving federated learning”. In 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), IEEE (2020): 410-419.
6. Zaixi Zhang., et al. “Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients”. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2022): 2545-2555.