2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR) | 978-1-7281-8243-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/AIPR50011.2020.9425094

RefinedFed: A Refining Algorithm for Federated Learning

Mohamed Gharibi
Dept. of Computer Science and Electrical Engineering
University of Missouri-Kansas City
Kansas City, MO, USA
mggvf@mail.umkc.edu

Praveen Rao
Dept. of Health Management and Informatics
Dept. of Electrical Engineering and Computer Science
University of Missouri-Columbia
Columbia, MO, USA
praveen.rao@missouri.edu

***Abstract*—Federated learning (FL) is a machine learning approach where the goal is to train a centralized model using a large number of clients that host private datasets. FL trains a smaller version of the model at each dataset site and then aggregates all the models at the server. In practice, clients (i.e., dataset holders) that participate in the learning process may possess corrupted or noisy datasets resulting in low accuracy models. Additionally, malicious clients may poison the data or carry out model discovery attacks.**

In this paper, we propose a refining algorithm called RefinedFed, to eliminate corrupted, low accuracy, and noisy models that can negatively impact the centralized model by reducing its accuracy or cause other malicious activities. Furthermore, RefinedFed reduces the uplink communication cost with the centralized server, which in return results in faster aggregation on the server side. Based on our preliminary experiments on the MNIST dataset, we observed that RefinedFed improved the global model accuracy from 84% to 91% while consuming less time for aggregation.

***Index Terms*—Privacy-preserving machine learning, federated learning, noisy data, malicious attacks**

I. Introduction

In 2016, McMahan et al. [1] proposed FL, which is a distributed machine learning approach to train a centralized/global model at a server using a large number of private datasets held by clients without obtaining the actual data. The key idea is to train a smaller version of the model at each client and then aggregate all the models at the server. The goal is to minimize the objective function [18] (Equation 1)

$$\min_{w} f(w) = \sum_{i=1}^{n} p_i F_i(w) = E_i[F_i(w)], \tag{1}$$

where n is the number of clients and $\sum_i p_i = 1$ s.t. $p_i \geq 0$. FL allows the training of models on distributed datasets without requiring the transfer of the actual datasets to the server. In particular, FL is one instance of the more general approach of “bringing the code to the data, instead of the data to the code”. As a result, we will train a model without having access to the actual datasets on the clients.

FL is one of the widely adopted techniques in the context of privacy-preserving machine learning in order to train a model on data that is not accessible such as patient records in hospitals. Instead of uploading the data to a centralized server where a model can be trained, FL techniques rely on sending the model to clients, which in return will train a model on their local data without having to share them with the server. Figure 1 shows a simplified architecture of FL. Furthermore, FL is often deployed to train models on data from edge and wearable devices, which continuously collect data from users such as smartphones and medical equipment. For example, FL is popularly used in providing smart services on smartphones. Google extensively uses FL in the Gboard mobile keyboard [9]–[12] and Android Messages [13]. Apple uses cross-device FL in iOS 13 [8]. The model that predicts the next word in a smartphone's keyboard was also trained using the FL technique.

In this paper, we propose a refining algorithm called RefinedFed, to eliminate corrupted, low accuracy, and noisy models during FL. The key contributions of our work are as follows:

- RefinedFed refines all clients models based on their accuracy after being tested locally at each client local dataset. When a model passes a specific accuracy threshold, it will be collected by the server. Otherwise, the model will be dropped and will not be collected in that specific round.
- RefinedFed reduces the uplink communication with the server, since many corrupted models, noisy models, and low accuracy models will not be collected by the server. This will save network bandwidth usage for the server.
- RefinedFed reduces the aggregation time on the server by eliminating low accuracy and corrupted models from the pool of models at every aggregation phase. Hence, the number of models that are included in the aggregation phase are less than the actual models that were included in the training phase.
- We evaluated RefinedFed on the MNIST dataset and observed that it yielded an accuracy of 91% compared to the traditional approach of FL, which yielded only 84% accuracy.

The rest of the paper is organized as follows: Section II discusses the background and related work on FL. Section III provides the motivation for our work. Section IV describes

Authorized licensed use limited to: University of Missouri Libraries. Downloaded on June 20,2025 at 20:44:04 UTC from IEEE Xplore. Restrictions apply.

our approach in detail. Section V reports our experimental evaluation. Finally, we conclude in Section VI.

II. Background and Related Work

The general description of FL was given by Bonawitz et al. [2]. Moreover, Konečný et al. [3], Bonawitz et al. [4], [6], and McMahanet et al. [5] discuss the theory behind FL in a more detailed way to address the fundamental problems of privacy, ownership, and locality of data. FL was initially introduced to target mobile and edge device applications [7]. Later on, FL was also used across multiple organizations such as hospitals. We will call these two settings "cross-device" and "cross-silo", respectively [8].

In both cross-device and cross-silo settings, the process starts when a server creates a model (global model) and shares it with all clients. At that point, each client will have the same version of the global model. (Once the model is on the client side, it is called a local model.) The training phase starts at each client with specific number of epochs among all clients. Once the training is complete, each client will send the updated weights to the server. The server will aggregate all of the weights and average them to produce a generalized global model. This is considered as one round of training for the global model. The produced model will then be sent to all of the clients that are considered in the next round. This process will continue to repeat and a more generalized model will be produced.

There are two major types of training algorithms in the FL infrastructure: synchronous and asynchronous. While the asynchronous algorithm was implemented earlier and have had much success [14], recently, Goyal et al. [15] and Smith et al. [16] changed the trend towards the synchronous batch training. McMahan at al. [1] proposed the algorithm of *FederatedAveraging* (FedAvg), which showed a huge success in the field, yet it still has some limitations such as dropping all the devices that fail to finish a specific number of epochs within a specific amount of time [18]. In general, FedAvg provides a way to filter the clients from being included in the aggregation. There is a list of requirements that each client has to fulfill in order to be included in the current round, such as having enough phone charge, having a stable Internet connection, ensuring the phone is not being heavily used, etc. These requirements make sure that the process will not affect the user usage. Furthermore, it also selects phones with good data and good Internet connectivity to avoid a client from being dropped during a training round.

Nishio et al. [17] have proposed a decentralized framework for training models while preserving their privacy. Their proposed protocol, FedCS, helps in solving the clients selection problem by managing clients based on their resources. FedCS is another protocol that allow client selection before the aggregation phase. For example, clients with poor Internet connection have to be managed in another way. However, such protocols are not selecting the clients based on the model accuracy when tested locally. Rather they are selecting and grouping the models that should be included in the current round before the training starts.

III. Motivation

Averaging all client models is the standard approach currently used to generate a global, generalized model with a better accuracy. This technique is similar to Random forests where the idea is to average all the over-fitted tree models to produce a better overall model. However, this approach faces a real challenge when participating entities (i.e., clients that participate in training small models on premises) do not hold "good" data or their data might include noise. For example, in the case of using FL for improving next-word predictions in smartphones, many use English language to type words in other languages (e.g., a person may type "salam" in English while it's a greeting in Arabic.) Not to mention the grammatical mistakes and shortcuts such as typing "u" instead of "you". Different accents and different slangs may also lower the model accuracy such as typing "goin" instead of "going" etc. The models that will be collected from such clients and will be harmful for the general model.

On the other hand, we have computer vision models that were trained on images. However, some clients may have a huge number of images with high resolution. Others may have only a few images, corrupted images, low resolution images, black and white images, or images with a lot of noise that will affect the overall model in a bad way. Moreover, collecting models from a much larger crowd requires more computation power, bandwidth, and introduces additional latency. Therefore, our proposed algorithm will run a simple accuracy test for each client model after each round and based on its accuracy, the model will be either included or excluded in further operations on the server.

Algorithm 1 $RefinedFed$: an extended algorithm for $FederatedAveraging$

```
1: Server executes:
2: initialize $w_0$
3: for each round $t$=1,2,... do
4:     Select K eligible clients to compute updates
5:     Wait for updates from K clients (indexed 1, . . . , K)
6:     Run clients on local testset
7:     if client accuracy less than threshold then
8:         drop client
9:     end if
10:    ($\Delta^k$, $n^k$) = ClientUpdate($w$) from client k $\in$ [K]
11:    $\bar{w}_t = \sum_k \Delta^k$        // Sum of weighted updates
12:    $\bar{n}_t = \sum_k n^k$        // Sum of weights
13:    $\Delta_t = \Delta_t^k / \bar{n}_t$        // Average update
14:    $w_{t+1} \leftarrow w_t + \Delta_t$
15: end for
```

IV. RefinedFed

RefinedFed is a protocol that has the goal of collecting the federating models that pass a certain test accuracy threshold

Authorized licensed use limited to: University of Missouri Libraries. Downloaded on June 20,2025 at 20:44:04 UTC from IEEE Xplore. Restrictions apply.

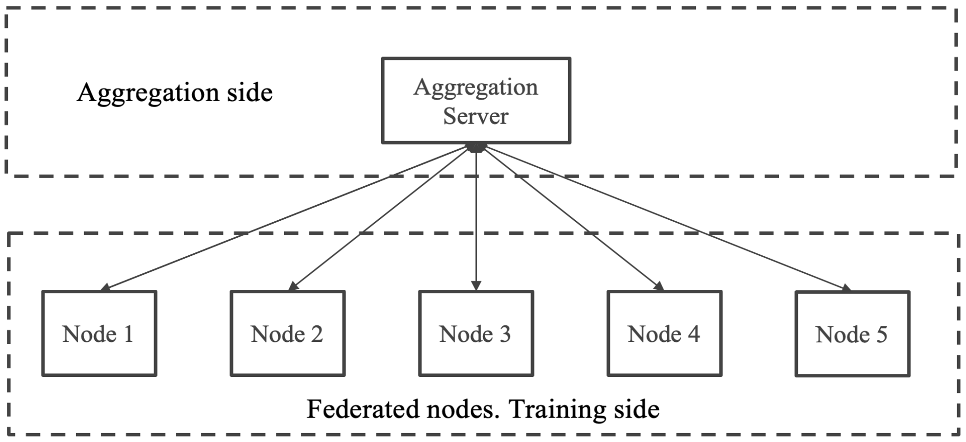

Fig. 1. The simplified architecture of FL where the server initially sends a global model to the clients. The clients perform local training and share updated weights with the server. The server aggregates the weights and updates the the global model and continues to perform these steps again.

after each round. In other words, it is a protocol to decide which model should be included in the server model aggregation process, such as averaging the weights while training the model. This protocol as shown in Algorithm 1 helps in building and training a more accurate model. Furthermore, when the model fails to pass a certain test accuracy, the model will not be collected and that will reduce the computation time and the bandwidth usage on the server which in return speeds up the aggregation phase on the server. See Figure 2. We evaluated RefinedFed on the MNIST dataset [19] and compared with the traditional FL approach. We would like to mention that our protocol is an extension of McMahan et al. [1] and Bonawitz et al. [6] with some modification to test each model on its local testing dataset and select those models that pass a certain threshold before sending the updates to the server. The second function in Algorithm 1, ClientUpdate, is the same as that in the original paper [1].

V. Experiments

We used the MNIST dataset [19] for all the experiments by equally distributing the number of the training images on all the clients. We used PyTorch and Pysyft platforms to simulate different number of clients with the centralized server. We set up a total of five clients. We also added Laplacian noise to the training images for few clients to simulate corrupted data, low-accuracy, and noisy models. By design, RefinedFed will not avoid using these models in the aggregation phase on the server. We chose Laplacian noise as the peak of the Laplace distribution is sharper than the Gaussian distribution. This implies that the number of Laplace samples around the zero are more than Gaussian. In practice, both Laplace and Gaussian noise perform well in terms of adding noise to the images and any other type of noise can be used as well. The number of noisy models are the same in both experiments for FL with and without RefinedFed.

Figure 3 shows the accuracy achieved on the server over 10 epochs. Clearly, RefinedFed achieved an higher accuracy than the traditional way of FL. While RefinedFed's accuracy reached 91%, the traditional approach achieved only 84%. Table I shows the accuracy achieved in each epoch.

VI. Conclusion

FL is distributed machine learning approach that focuses on sending the model to the data on the clients while preserving the privacy of the actual data. This approach is possible by training large number of models on the clients followed by aggregating them using different available algorithms (e.g., FedAvg) in order to generate a generalized global model. However, during the aggregation, low accuracy or defective models may affect the overall model accuracy. Therefore, we introduced RefinedFed, a simple, yet effective approach to filter such models and generate a better global model. Based on our experiments using MNIST, we observed that our approach outperformed the standard FL approach that does not use any filtering mechanisms. While RefinedFed achieved 91% accuracy, the standard FL approach could achieve only 84% accuracy.

Acknowledgment

We would like to acknowledge the partial support of NSF Grant No. 1747751.

References

[1] B. McMahan, E. Moore, D. Ramage, and et al. "Communication-efficient learning of deep networks from decentralized data," In Artificial Intelligence and Statistics, pp. 1273-1282. PMLR, 2017.

[2] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, B. H. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. "Practical secure aggregation for privacy-preserving machine learning," In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1175-1191. 2017.

[3] J. Konečný, B. H. McMahan, D. Ramage, and P. Richtárik. "Federated optimization: Distributed machine learning for on-device intelligence," arXiv preprint arXiv:1610.02527 (2016).

[4] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, B. H. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. "Practical secure aggregation for privacy-preserving machine learning," In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1175–1191. ACM, 2017.

Authorized licensed use limited to: University of Missouri Libraries. Downloaded on June 20,2025 at 20:44:04 UTC from IEEE Xplore. Restrictions apply.

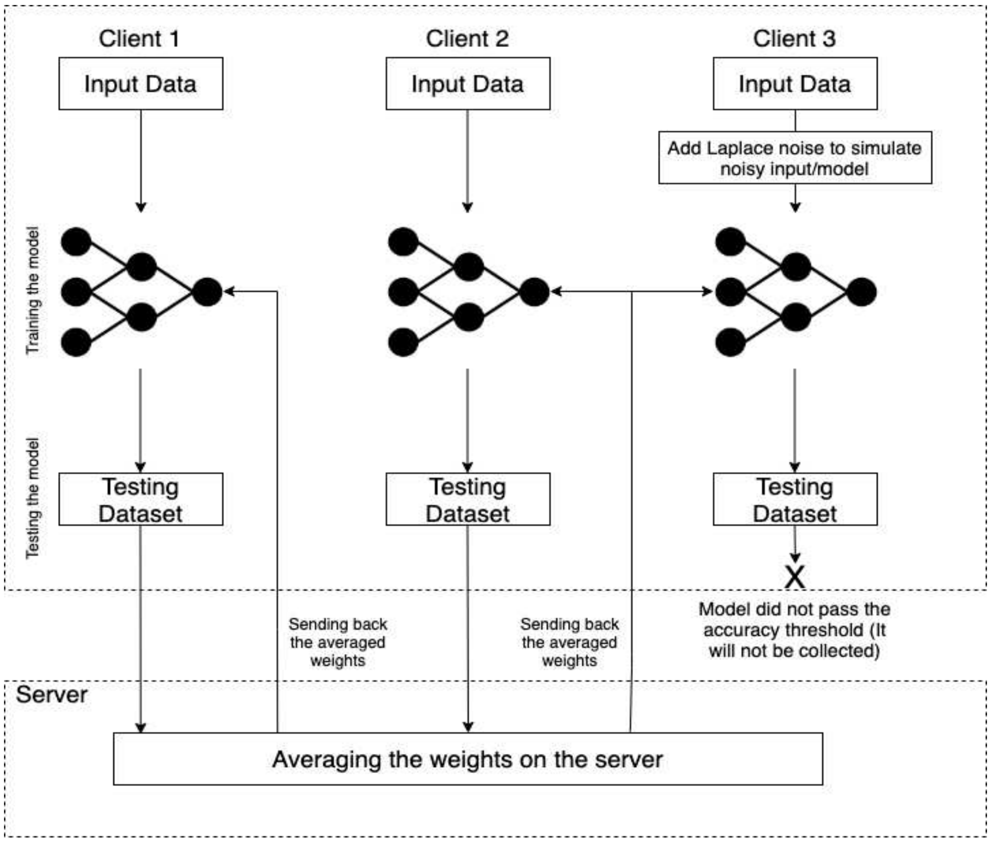
Client 1
Client 2
Client 3
Input Data
Input Data
Input Data
Add Laplace noise to simulate noisy input/model
Training the model
Testing the model
Testing Dataset
Testing Dataset
Testing Dataset
X
Model did not pass the accuracy threshold (It will not be collected)
Sending back the averaged weights
Sending back the averaged weights
Server
Averaging the weights on the server

Fig. 2. RefinedFed Architecture. A local testing dataset will be added to each client to test the model before the collecting phase. Models pass a certin accuracy threshold will be collected by the server. Otherwise, the model will be dropped.

TABLE I
ACCURACY OF FL OVER 10 EPOCHS WITH AND WITHOUT REFINEDFED

	Accuracy over 10 epochs									
Epoch number	1	2	3	4	5	6	7	8	9	10
FL	11%	11%	22%	47%	51%	49%	64%	77%	82%	84%
RefinedFed	11%	30%	36%	43%	62%	79%	83%	86%	88%	91%

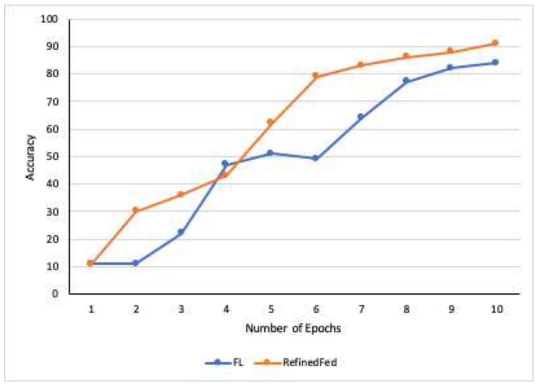

Fig. 3. The accuracy over 10 epochs is shown: FL vs. RefinedFed

[5] B. H. McMahan, D. Ramage, K. Talwar, and L. Zhang. "Learning differentially private recurrent language models," arXiv preprint arXiv:1710.06963 (2017).

[6] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, and et al. "Towards federated learning at scale: System design," arXiv preprint arXiv:1902.01046 (2019).

[7] B. H. McMahan and D. Ramage. "Federated learning: Collaborative machine learning without centralized training data," April 2017. URL https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. Google AI Blog.

[8] P. Kairouz, B. H. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, and et al. "Advances and open problems in federated learning," arXiv preprint arXiv:1912.04977 (2019).

[9] S. Pichai. "Google's Sundar Pichai: Privacy Should Not Be a Luxury Good," New York Times, May 7, 2019.

[10] 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 1811.03604, 2018.

[11] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays. "Applied federated learning: Improving Google

Authorized licensed use limited to: University of Missouri Libraries. Downloaded on June 20,2025 at 20:44:04 UTC from IEEE Xplore. Restrictions apply.

keyboard query suggestions," arXiv preprint 1812.02903, 2018.
[12] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays. "Federated learning of out-ofvocabulary words," arXiv preprint 1903.10635, 2019. URL http://arxiv.org/abs/1903.10635.
[13] support.google. Your chats stay private while Messages improves suggestions, 2019. URL https://support.google.com/messages/answer/9327902. Retrieved Aug 2019.
[14] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, and et al. "Large scale distributed deep networks," In Advances in neural information processing systems, pp. 1223-1231. 2012.
[15] P. Goyal, P. Dollar, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. "Accurate, large minibatch SGD: Training imagenet in 1 hour," arXiv preprint arXiv:1706.02677, 2017.
[16] S. Smith, P. J. Kindermans, C. Ying, and Q. V. Le. "Don't decay the learning rate, increase the batch size," In International Conference on Learning Representations (ICLR), 2018.
[17] T. Nishio, and R. Yonetani. "Client selection for federated learning with heterogeneous resources in mobile edge," In ICC 2019-2019 IEEE International Conference on Communications (ICC), pp. 1-7. IEEE, 2019.
[18] T. Li, Z. Zaheer, S. Maziar, T. Ameet, S. Virginia. "Federated optimization in heterogeneous networks," arXiv preprint arXiv:1812.06127, 2018.
[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.

Authorized licensed use limited to: University of Missouri Libraries. Downloaded on June 20,2025 at 20:44:04 UTC from IEEE Xplore. Restrictions apply.