Efficient Federated Learning with Self-Regulating Clients

Zahidur Talukder
University of Texas at Arlington

Mohammad A. Islam
University of Texas at Arlington

1. INTRODUCTION

Motivation. Since its inception [1], Federated Learning (FL) has been enjoying a strong interest from the privacy-preserving AI research community. FL also has been commercially implemented in popular applications such as Google's Gboard and Apple Siri. In FL, Machine Learning (ML) models are trained cooperatively by many clients (e.g., mobile phones) without explicitly sharing their identity and private data. The training is done through iterative communication rounds between a central parameter/model server and clients. In each communication round, the global model is supplied to a subset of clients who use their private data to do local training on their respective devices. The updated models from each participating client are anonymously sent back to the central server and aggregated to get the global model for the next communication round.

Due to its siloed decentralized training on locally generated client data, FL suffers from data heterogeneity where the statistical differences (i.e., non-IID) between the training samples for local model updates cause model drifting during aggregation at the central server and affect convergence. The data heterogeneity is prevalent in FL as different clients collect the training data from different data sources using different hardware/sensors. FL clients, therefore, have a varying degree of reliability in terms of their data quality. They can even be malicious. However, the aggregation approach proposed in [1], known as Federated Averaging or FedAvg, does not directly address data heterogeneity. FedAvg uses a weighted averaging where clients get weights proportional to the size of their respective training data set.

Limitations of existing approaches. A prominent line of work to circumvent the data heterogeneity issues focuses on controlling the aggregation weights at the central server [2, 3, 4]. The foundational idea here is to put a lower weight on the updates that are "unfriendly" for the central model. However, for the clients receiving a low model aggregation weights, and hence, contributing little to the central model, the computation cost of local model training and the communication cost of sending the model update to the central server can be perceived as a "waste of resources", especially since clients' devices in FL setup are typically considered to be resource-constrained.

Our contribution. Our goal in this work is to avoid the aforementioned resource waste by stopping the clients from participating in model update rounds if their updates are destined to make little to no impact. More importantly, we want the clients themselves to be able to anticipate this type of resource waste and refrain from participating in the model updates (in the FL rounds they are selected). We propose a novel FL approach, **Fed**erated Learning with **S**elf-**R**egulating **C**lients (FedSRC) where clients implement checkpoints along their local training path to determine whether they should exit from participation. FedSRC's design is motivated by our empirical observation that, similar to reducing aggregation weights, discarding clients with "bad data" improves the global model (Fig. 2). We also observe that the clients with bad data suffer from worse model accuracy than others. Based on these observations, in FedSRC, client checkpoints are designed to evaluate the model accuracy and compare with the previous round's median training accuracy provided by the central server. The overhead of FedSRC's checkpoints implementation is low as they only require lightweight inference on a small data set. Also, at the end of each update round, participating clients need to send only one additional parameter, training accuracy, to the central server. Fig. 1 illustrates the working principle of FedSRC.

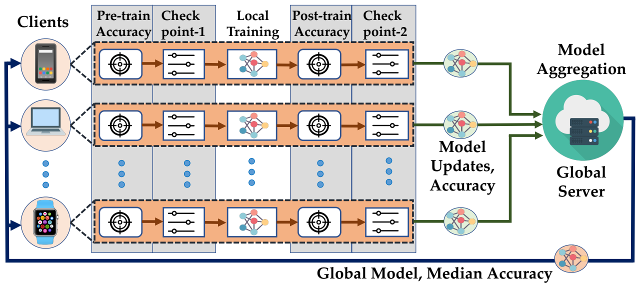


Figure 1: FedSRC adds two checkpoints based on pre-train and post-train model accuracy to enable self-regulated client participation.

We evaluate FedSRC using three different data sets. We show that with the same number of communication rounds, FedSRC can save as much as 39% on the communication and 37% on the computation.

Merits of FedSRC. By design, FedSRC can improve the overall communication, and computation-efficiency of FL with weighted model aggregations [1, 2, 3, 4]. More importantly, FedSRC can work seamlessly with other communication efficient FL approaches such as model compression and structured update [5]. FedSRC can also be integrated with techniques that tackle model and data poisoning attacks at the central server. In fact, FedSRC can assist thwarting data poisoning attacks where the client's FL steps (i.e., client firmware) are not compromised. In any case, FedSRC does not introduce any new attack vector and performs no worse than existing approaches.

Parallels with client selection in FL. FedSRC's approach has some similarities with active client selection [6, 7]. However, FedSRC is fundamentally different from client selection where the central server creates client profiles to select the suitable clients in each communication round. Client selection approaches need to tag client updates with their client IDs. Therefore, they cannot maintain client anonymity diminishing FL's privacy. In FedSRC, on the other hand, all client updates are sent to the central server anonymously following standard FL protocols.

2. BACKGROUND

2.1 Federated Learning (FedAVG)

We here formalize FL in the context of neural network based ML system. A neural network can be represented as a function $f(x, \Theta) = y$ that maps input x to an output y, where $\Theta \in \mathbb{R}^k$ is the parameters of f. An observed pair $\langle x, y \rangle$ represents a training sample and a training set with m samples is a collection $D = \{\langle x_i, y_i \rangle \ i = 1, \cdots, m\}$. The loss function on a training set $\mathcal{L}_f(D, \Theta) = \frac{1}{|D|} \sum_{\langle x_i, y_i \rangle \in D} l_f(x_i, y_i, \Theta)$ where $l_f(x_i, y_i, \Theta)$ is the loss of prediction for sample $\langle x_i, y_i \rangle$ with model parameter Θ. A popular choice for the loss function is squared L2 norm, i.e., $l_f(x_i, y_i, \Theta) = (y_i - f(x_i, \Theta))^2$. The training objective here is to tune Θ to minimize $\mathcal{L}_f(D, \Theta)$.

In FL setting, each user $u \in \mathcal{U}$ holds a private subset of training sample D_u where $D = \cup_{u \in \mathcal{U}} D_u$. The loss function can be rewritten as $\mathcal{L}(D, \Theta) = \sum_{u \in \mathcal{U}} \frac{|D_u|}{|D|} \mathcal{L}_f(D_u, \Theta)$ where $\mathcal{L}_f(D_u, \Theta) = \frac{1}{|D_u|} \sum_{\langle x_i, y_i \rangle \in D_u} l_f(x_i, y_i, \Theta)$. To run stochastic gradient descent at communication round k, a random subset of users $\mathcal{U}^k \subset \mathcal{U}$ is chosen to form a minibatch $B^k = \cup_{u \in \mathcal{U}^k} D_u$. In practice, $|\mathcal{U}^k| \ll \mathcal{U}$ and a subset of user's local samples D_u may be chosen for B^k. With this the loss gradient $\Delta \mathcal{L}_f(B^k, \Theta^k) = \frac{1}{|B^k|} \sum_{u \in \mathcal{U}^k} \delta_u^k$ where $\delta_u^k = |D_u| \Delta \mathcal{L}_f(D_u, \Theta^k)$. With a learning rate of η the gradient descent step at the central model server is

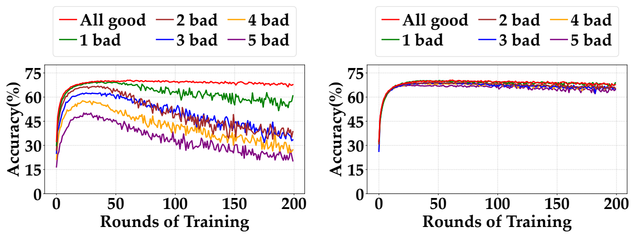
$$\Theta^{k+1} \leftarrow \Theta^k - \eta \frac{\sum_{u \in \mathcal{U}^k} \delta_u^k}{\sum_{u \in \mathcal{U}^k} |D_u|}, \qquad (1)$$

which only requires $\langle |D_u|, \delta_u^k \rangle$ from each user to be sent to the central server. After each minibatch training, the updated model Θ_{k+1} is synchronized with all users and then a new round of minibatch update is commenced with a new random set of users. Hence, each user is responsible (in the minibatch rounds it is chosen for gradient descent update) for calculating the loss gradient δ_{tu} using their local data. Data privacy is greatly enhanced in FL since raw user data never leaves the user devices. Moreover, Eqn. (1) does not require user identity and hence users can send their updates $(\langle |D_u|, \delta_u^k \rangle)$ anonymously without any meta-data.

2.2 Motivating Example

As a foundation of our approach, we hypothesize that participation from unreliable clients with bad data quality is harmful to the overall accuracy and convergence in the central model aggregation. To support our hypothesis, we run experiments on the CIFAR10 data set with 10 clients. We vary the number of clients with corrupted data and check the accuracy of the global model after each communication round. We consider all data of an unreliable client are mislabeled. For the purpose of better illustrating, here we consider clients with 100% data corruption to amplify the impact of clients' data quality on global model's performance.

Fig. 2(a) shows the impact of bad clients on the global accuracy. As the number of bad clients increases, accuracy and convergence greatly suffer. Next, in Fig. 2(b) we show the global accuracy where we discard (i.e., set weights to zero) the bad clients from central model aggregation. Here, we see that the global accuracy and convergence is not adversely affected if we can identify and discard bad clients from model updates, thereby supporting our hypothesis that



(a) FedAVG (b) Client discarding

Figure 2: (a) Accuracy of the global model with different number of unreliable clients with bad data. (b) Impact of discarding bad client on the global model.

filtering out imperfect clients helps the global model. Moreover, we also identify that bad clients suffer from low model accuracy due to their bad data quality.

3. FedSRC

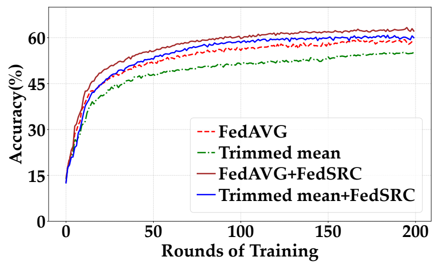
Here we present FedSRC where clients self-regulate their participation in model updates towards improved communication and computation efficiency without sacrificing, in some cases improving, the accuracy of the global model.

Working principle. In FedSRC, after being selected by the central model server, a client determines whether to participate in the model update based on two checkpoints placed along its processing path. The first checkpoint (*Checkpoint-1*) is after the client receives the updated model from the central server. In *Checkpoint-1*, the client determines whether to proceed with local training or remove itself from the model update of the current round. The second checkpoint (*Checkpoint-2*) is after the local training is completed. In *Checkpoint-2*, the client determines if the model update should be sent to the central server or not. If a client exits from an update round in *Checkpoint-1*, it saves the computation for local training and communication to send the update to the central server. On the other hand, a client exiting the model update at *Checkpoint-2* only saves on the communication. Next, we describe how the two checkpoints work.

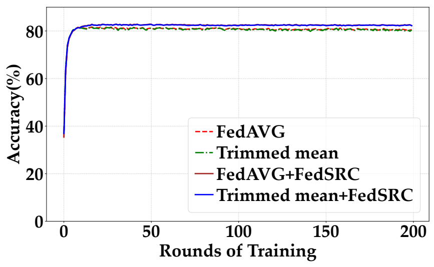
Added steps for checkpoints implementation. For our checkpoints implementation, we add two additional steps on the client where a test data set $D_u^k \subset D_u$ is used to determine the model accuracy before and after the local training. With central model Θ^k the pre-train accuracy is $A_{u,pre}^k = \mathcal{A}(D_u^k, \Theta^k)$, where $\mathcal{A}(D_u^k, \Theta^k)$ calculates the accuracy of model Θ^k using test set D_u^k. The post-train accuracy is $A_{u,post}^k = \mathcal{A}(D_u^k, \Theta_u^{k+1})$, where $\Theta_u^{k+1} = \Theta^k - \eta \frac{\delta_u^k}{|D_u|}$.

In addition, during model update, clients also report the post-train accuracy $A_{u,post}^k$ to the central server. The central server determines the median post-train accuracy $\tilde{A}_{post}^k = \text{Med}_{u \in \mathcal{U}^k}(A_{u,post}^k)$ and sends it to the clients selected for the next round along with the updated model Θ^k.

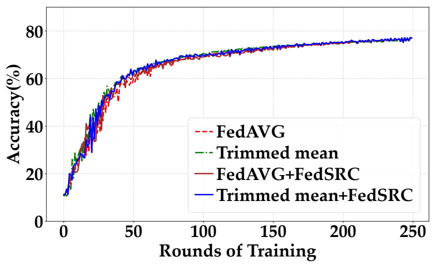
Client checkpoints for self-regulation. In *Checkpoint-1*, pre-train accuracy $A_{u,pre}^k$ is compared against the previous round's median post-train accuracy $\tilde{A}_{post}^{k-1}$. A client moves on to local training if $A_{u,pre}^k > \tilde{A}_{post}^{k-1} - \alpha$, where α is a FedSRC parameter that controls how aggressively clients regulate themselves. After local training, *Checkpoint-2* allows model update $\langle |D_u|, \delta_u^k, A_{u,post}^k \rangle$ to be sent to the central server if $\text{Abs}(A_{u,pre}^k - A_{u,post}^k) > \beta$, where β is another FedSRC parameter regulating *Checkpoint-2*'s effectiveness.



(a) CIFAR10, $\alpha = 0.05, \beta = 0.15$

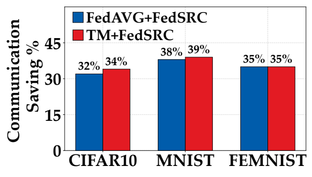


(b) MNIST, $\alpha = 0.05, \beta = 0.15$

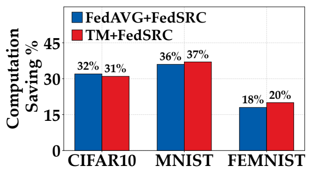


(c) FEMNIST, $\alpha = \sigma$, $\beta = 2\sigma$

Figure 3: FedSRC performs equally or better than existing FL approaches with communication and computation savings.

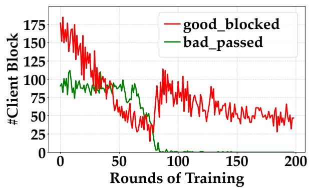


(a) Communication

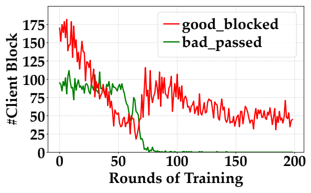


(b) Computation

Figure 4: FedSRC's saving compared to FedAVG.



(a) FedAVG+FedSRC



(b) TM+FedSRC

Figure 5: Number of good client blocked and bad client pass by our fileter

4. EVALUATION

Experimental setup. We evaluate FedSRC using three commonly used data sets with FL - CIFAR10/ 60,000/ 10,000/ 100, MNIST/ 50,000/ 10,000/ 100, and FEMNIST/ 341,873/ 40,832/ 3383 where the data set configuration is represented as Name/ #training sample/ #test sample/ #clients. For CIFAR10, we use one block VGG network, and for MNIST and FEMNIST, we use a logistic regression classifier with Tensorflow and Keras sequential model. We use small ML models considering that in FL setup the client devices are typically resource-constrained. We use two aggregation weight-based benchmark algorithms - FedAvg and Trimmed Mean (TM). We implement FedSRC on top of these two algorithms - FedAvg+ FedSRC and TM+ FedSRC to see how much benefit FedSRC brings to the existing FL approaches. We consider 30% clients of each data set suffer from bad data quality with 30% added Gaussian noise.

Execution of FedSRC. During the evaluation, we identify that the checkpoints in FedSRC need to be enabled after a few tens of communication rounds to allow a stable median accuracy $\tilde{A}^{k-1}_{post}$. We wait for 10 rounds for CIFAR10 and MNIST, and 50 rounds for FEMNIST. We use $\alpha = 0.05$ and $\beta = 0.15$ for CIFAR10 and MNIST, while for FEMNIST we use $\alpha = \sigma^k$ and $\beta = 2\sigma^k$ where $\sigma^k = \text{Std}_{u \in \mathcal{U}^k}(A^k_{u,post})$.

Results. Fig. 3 shows the accuracy of the global model as we progress through communication rounds for the four different algorithms. We see that for CIFAR10 (Fig. 3(a)) and MNIST (Fig. 3(b), FedSRC results in better accuracy. Meanwhile, for FEMNIST, FedSRC retains a similar performance as FedAvg and TM.

Fig, 4 shows FedSRC's savings compared to FedAVG (which is the same at TM) where there is no client-side control is implemented. We calculate the savings based on how many communication and computation events the clients have averted using FedSRC compares to FedAVG. In Fig. 4(a), we see more than 30% communication savings across all data sets. Meanwhile, Fig. 4(b) shows that CIFAR10 and MINST have computation savings comparable to the communication savings. For FEMNIST, however, we see a dip in computation saving which indicates that for FEMNIST many client updates crossed *Checkpoint-1*, but got stopped at *Checkpoint-2*. Whereas for CIFAR10 and MNIST, their similar communication and computation savings indicate the updates that passed *Checkpoint-1*, also passed *Checkpoint-2* in most cases.

5. CONCLUDING REMARKS

In this paper, we presented our preliminary finding on a novel approach, FedSRC, for improving communication and computation efficiency of FL without sacrificing the global model accuracy and client anonymity. To the best of our knowledge, this is the first attempt to regulate client participation from the client side.

Future work. With the motivating preliminary results, we plan to improve FedSRC's current implementation along with the following directions. *Convergence analysis.* We will conduct convergence analysis on our model's accuracy-based approach to establish theoretical performance guarantees for FedSRC. *Reducing checkpoint overhead.* Instead of a separate inference step to calculate model accuracy for FedSRC's checkpoints, we plan to utilize the calculation done in the forward pass during model training to extract the model accuracy with minimal overhead. *Parameter sensitivity.* We will study the impact of checkpoint parameters α and β on FedSRC. We also plan to implement auto-tuning for these two parameters. *Extended evaluation.* Finally, we will also extend our evaluation to more data sets and compare FedSRC with various benchmark algorithms, especially with active client selection algorithms.

6. REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[2] S. Li, E. Ngai, F. Ye, and T. Voigt, "Auto-weighted robust federated learning with corrupted data sources," *ACM Trans. Intell. Syst. Technol.*, feb 2022.

[3] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *ICML*, 2020.

[4] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, “Byzantine-robust distributed learning: Towards optimal statistical rates,” in *ICML*, 2018.

[5] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” *arXiv preprint arXiv:1610.05492*, 2016.

[6] Y. J. Cho, J. Wang, and G. Joshi, “Client selection in federated learning: Convergence analysis and power-of-choice selection strategies,” *AISTATS*, 2022.

[7] J. Goetz, K. Malik, D. Bui, S. Moon, H. Liu, and A. Kumar, “Active federated learning,” *arXiv preprint arXiv:1909.12641*, 2019.