Federated Learning Using Deep Networks on a Distributed and Decentralized System

Akul Agrawal, Divya D Kulkarni, and Shivashankar B. Nair

Indian Institute of Technology Guwahati, Guwahati, Assam, India
{akulagrawal, divyadk, sbnair}@iitg.ac.in

Abstract. With a massive amount of data collected from the ever increasing devices and applications in the past few years, it is now possible to deploy machine learning techniques to improve user experience. To tackle the problem of user privacy and huge bandwidth requirement in sending the user data to the company servers to train machine learning models, Google introduced Federated Learning (FL) [1] in 2016. FL is a distributed implementation of machine learning which enables on-device training of the models. The state-of-the-art FL model uses the FederatedAveraging (`FedAvg`) algorithm [1]. In this approach, in each communication round (CR), every end device (client) trains the machine learning model and a fraction of clients (F_{Cl}) sends the trained weights to a common server (cloud), which averages the obtained weights, and sends the averaged weights back to every client. The clients then use these averaged weights as initial state for the next communication round. This paper presents the effect of changing F_{Cl} on the performance of `FedAvg`. A preliminary study is conducted on the FL setup with the MNIST Digit dataset. The results conclude that there exists an optimal fraction of clients, keeping rest of the hyper parameters same, such that the number of communication rounds required to achieve a certain accuracy is minimum.

Keywords: Federated Learning · `FedAvg` algorithm · Optimal fraction of clients.

1 Introduction

In the past few years, several large concerns such as Google, Microsoft and Amazon have been actively devising machine learning techniques and using them in a gamut of applications. The process entails assimilating a huge amount of data so as to train the machine learning models. A major portion of such data is pulled up from users who consume their products. This in turn rakes up the issue of data privacy as it could include data that is user-specific or personal. For instance, text messages and recent searches may be treated as classified by a user who in turn may not wish them to be known by a third party. Processing huge amounts of data also requires a humongous amount of centralized or cloud based computing resources which naturally increases cost. Most often the entire

collected data is sent to a centralized server, that runs the machine learning model. The trained models are then sent to each of the users to enhance their experience. In the process, all user-specific or private data incurs the risk of being leaked from such centralized systems.
Federated Learning (FL) is a decentralized training approach that endeavours to circumvent this issue. FL allows end devices owned by the users to run the machine learning model within them locally. This avoids sending of private or sensitive data to the centralized servers. FL has the advantage that the model obtained using this approach gives better performance for user-specific data as compared to other traditional approaches [4].
In the ideal FL based approach, separate machine learning models are trained concurrently on each of the end devices using the data made available to it by the user on that device. This means that the end devices also need to possess a significant amount of computational capability. This issue has been addressed to quite an extent due to the advent of smart phones with considerable amount of memory and powerful processors that even cater to AI. This paradigm shift in running machine learning models on centralized servers to end-user devices is expected to improve the efficiency and performance of data-driven products such as hyper-personalized recommendation engines, e-commerce pricing engines, etc. An FL approach also provides a quicker response to fast-changing consumer behaviours, while also bringing down the cost of computation. It removes the bound on both the computational power and the data accessible to any industrial company by the training model on end devices. These bounds curtail the performance of traditional centralized approaches to implement machine learning. In this paper, we define the hyper parameters required to be adjusted for an efficient implementation of FL, and analyze the effect of variation in these hyper parameters on the FL system to decide how to adjust these hyper parameters. The `FedAvg` algorithm [1], which is the most commonly used algorithm to implement FL, is described as follows. Subsequent part describes the working of the FL and gives a brief intuition of how the `FedAvg` algorithm works.

The FL approach using `FedAvg` algorithm mainly includes the following steps in the following order:

1. A machine learning model is defined in a cloud.
2. The end devices download this model.
3. Each device trains the model locally for a fixed number of epochs on the data stored in that device which is specific to the user.
4. On completion, a fraction of the end devices upload an update [1] of the generated model to the cloud. This includes the resulting trained weights of their local models.
5. The cloud averages all the weights of all the received models and sends the resulting average weights back to each user.
6. Each user now starts with the These averaged weights obtained by the user form the new model. This model is used to for training locally.
7. The steps 3,4,5 and 6 together constitute a single communication round [1]. The communication rounds keep repeating and eventually, the accuracy of the local model improves.

The local model on any device can be used as soon as the first round of training is completed on that device. The model eventually improves with time. Thus, FL provides faster models as compared to the traditional approach. Since only model parameters are sent to the centralized cloud, the FL based approach provides lower latency as compared to the traditional approach wherein all the data needs to be sent. Further, since the model is trained on local devices, it is expected that better and more personalized predictions are made as compared to those generated by the centralized model. Care needs to be taken to schedule the implementation of the training so that it takes place only when the device is plugged in, idle and on an Internet connection so as to ensure that there is no impact on the performance of the end-user device.

2 Implementation Issues in FL

The `FedAvg` algorithm [1] has been described in the section 1 wherein the centralized server averages the weights it receives from the devices. It facilitates devices to perform multiple batch updates on local data and exchange the updated weights. In this paper, we focus on mainly 3 hyper parameters:

1. F_{Cl}: Fraction of clients sending updates to the cloud in every communication round.
2. CR: Number of communication rounds.
3. *Epochs*: Number of epochs for which a local model is trained in every client in a single communication round.

As is the case with many algorithms, the fixing of the values of the related hyper parameters is non-trivial. The `FedAvg` algorithm does not specify any particular method to decide on the values of these hyper parameters. Deciding these values remains one of the biggest challenges in the implementation of FL. In this paper, we attempt to empirically portray that there exists an optimal value for F_{Cl} such that CR required to achieve a given accuracy, is minimum. We also try to show that there exists an optimal value of CR at which the `FedAvg` can be stopped, without affecting the resulting accuracy.

3 Experimental Setup

We have used the MNIST Digit dataset [9], which was also used in the baseline `FedAvg` paper [1]. The data was partitioned over clients in a Non-Independent and Non-Identical manner, termed as Non-IID. To ensure this, the data was first sorted based on the digit label and divided into 200 shards of size 300. 100 clients were then assigned 2 shards each. Since most clients had only examples of at most two digits, this constituted to be a non-IID partition of the data.

3.1 Local Model

A CNN with two 5x5 convolution layers was used. The first one had 32 channels while the second had 64 channels, each followed by 2x2 max pooling. This was followed by a fully connected layer with 512 units and ReLu activation, and a final softmax output layer (1,663,370 total parameters) [1].

3.2 Setup

An FL system was emulated by implementing the `FedAvg` algorithm on a single system, with specifications as mentioned earlier. The total number of clients was taken to be 100 for all the experiments.

4 Experiments and Results

4.1 Subexperiment 1

This part of the experiment targets to analyze the effect of varying F_{Cl}, CR and *Epochs* on the accuracy of the FL system. We define the average training accuracy (A_{Tr}) as the average of the accuracies of the trained models in the F_{Cl} clients involved in sending updates in every communication round. The average training loss (L_{Tr}) is defined similarly by substituting loss functions in place of accuracies. For each setup, two graphs were plotted viz.

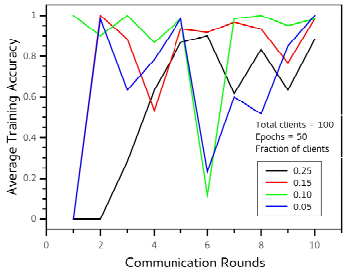
1. Average Training Accuracy vs. Communication Rounds and
2. Average Training Loss vs. Communication Rounds

From Fig. 1b and Fig. 1d, it can be observed that L_{Tr} seems to converge to -1 after 8 communication rounds when the number of *Epochs* is 10 and after 7 communication rounds in case of 50 *Epochs*. Thus, it can be inferred that with a lesser number of local epochs, more communication rounds may be required to achieve a similar accuracy.

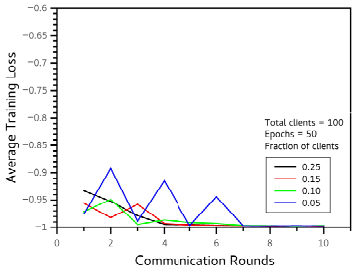
From Fig. 1a - Fig. 1d, it can be observed that when the value of F_{Cl} is 0.05, both the A_{Tr} and L_{Tr} curves fluctuate in the initial communication rounds. Thus, when the fraction of clients is very low ,the FL model seems to take more communication rounds to become stable and consistent while trying to achieve a fair accuracy .

In Fig. 1, since all the plots with $F_{Cl} = 0.15$ and $F_{Cl} = 0.25$ are similar, to build an efficient model, there could be an upper limit to the F_{Cl} value, beyond which, there is negligible improvement in the model accuracy.

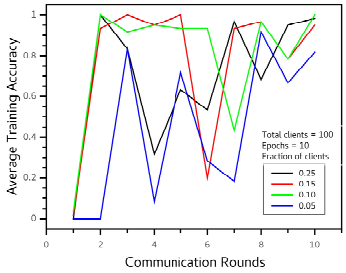
From Fig. 1e and Fig. 1f, it can be deduced that as the number of communication rounds increases, the average training accuracy and loss seem to become constant. This could aid in decreasing the frequency of sending updates to the server and decreasing the number of communication rounds, thereby decreasing time, energy consumption, and cost.



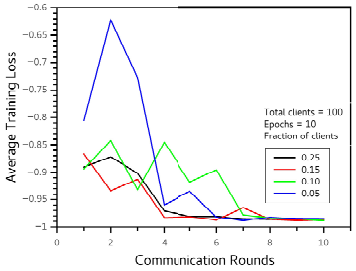
(a) A_{Tr} vs CR up up to CR=10, $Epochs$=50



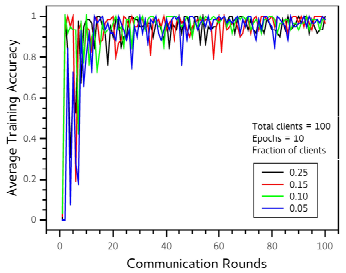
(b) L_{Tr} vs CR up to CR=10, $Epochs$=50



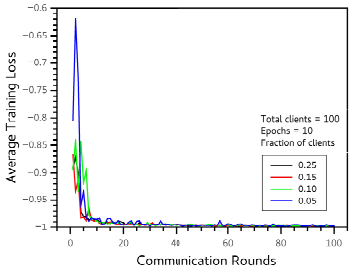
(c) A_{Tr} vs CR up to CR=10, $Epochs$=10



(d) L_{Tr} vs CR up to CR=10, $Epochs$=50



(e) A_{Tr} vs CR up to CR=100, $Epochs$=10



(f) L_{Tr} vs CR up to CR=100, $Epochs$=10

Fig. 1: The summary of experimental results conducted by varying the fractions of clients F_{Cl}, the number of communication rounds CR, and the local model training epochs $Epochs$. The average training accuracy (A_{Tr}) vs CR curves are plotted on the left and the average training loss (L_{Tr}) vs CR curves are plotted on the right.

4.2 Subexperiment 2

In this part, we aim to analyze the dependency between F_{Cl} and minimum value of CR, $minCR$, required to achieve a certain accuracy level, A_L. Since A_{Tr}, as defined in Subexperiment 1, keeps on fluctuating with CR (Fig. 1), using A_{Tr} as the accuracy level is expected to give incorrect results. Thus, to minimize the effect of fluctuation, we define the accuracy of this subexperiment as the moving average of w most recent accuracies, MA_w, with a constraint that the moving

standard deviation (σ) of the same w accuracies, σ_w, is bounded by a fixed value, σ_{THRESH}. Precisely,

$$\begin{aligned} minCR &= \min(CR) \\ \text{such that } MA_w &> A_L \\ \text{subject to } \sigma_w &< \sigma_{THRESH} \end{aligned} \tag{1}$$

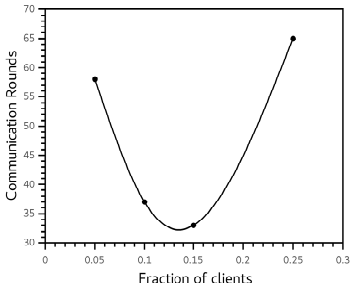


Fig. 2: $minCR$ vs F_{Cl} curve with $A_L = 0.97$ and $\sigma_{THRESH} = 0.03$ for the FL system with $Epochs = 10$ and Total clients = 100.

The curve appears to be parabolic in nature. It can be seen that to implement FL efficiently on a fixed sized dataset, there exists an optimal value of F_{Cl} which can achieve a given accuracy with the least CR as compared to all other values of F_{Cl}. This trend can be explained as follows. As we increase F_{Cl} from a very low value to a higher value, the total number of epochs across all clients before a communication round also increases. Thus, since more learning has been achieved due to the increasing number of epochs, a given accuracy (97% in our case) is attained with a lower number of communication rounds. This trend is seen prominently in the former half of the graph depicted in Fig. 2.

At around a value of F_{Cl} less than 0.15, the value of CR is minimum. Beyond this value, an increase in F_{Cl} would mean a decrease in the size of the data subset available to each client. Under such conditions, more epochs may result in overfitting at each client. The learning trend would thus become slower and require more communication rounds to generalize across the several data subsets. This results in a parabolic increase in the number of communication rounds required to achieve the given accuracy. This trend can be observed in the latter part of the graph in Fig. 1. The graph thus reveals the existence of an optimal value of F_{Cl} that reduces the number of communication rounds, given a target accuracy and dataset.

5 Conclusions and Future Work

This work presents the variation of different hyper parameters, which are required to be adjusted for an efficient implementation of any FL system. The empirical results presented reveal that there exists an optimal value of F_{Cl}, which can aid the system achieve a given accuracy in minimum time. Knowledge of this parameter, can thus greatly reduce time, cost and bandwidth consumed by excessive communication rounds. The training in an FL system eventually becomes very slow and could even saturate. Facilitating an adaptive control over the number of communication rounds based on accuracy and F_{Cl} could drastically bring down both cost and time.

References

1. McMahan, H B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA. pp. 1273–1282 (2017), http://proceedings.mlr.press/v54/mcmahan17a
2. Konečnỳ, J., McMahan, H B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated Learning: Strategies for Improving Communication Efficiency. CoRR **abs/1610.05492**, (2016). http://arxiv.org/abs/1610.05492
3. Konečnỳ, J., McMahan, H B., Ramage, D., Richtárik, P.: Federated Optimization: Distributed Machine Learning for On-Device Intelligence. CoRR **abs/1610.02527**, (2016). http://arxiv.org/abs/1610.02527
4. Anelli, V. W., Deldjoo, Y., Di Noia, T., Ferrara, A. : Towards Effective Device-Aware Federated Learning. CoRR **abs/1908.07420**, (2019). http://arxiv.org/abs/1908.07420
5. Zhao, Y. and Li, M., Lai, L., Suda, N., Civin, D., Chandra, V. : Federated Learning with Non-IID Data. CoRR **abs/1806.00582**, (2018). http://arxiv.org/abs/1806.00582
6. Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H B., Van Overveldt, T., David, P., Ramage, D., Roselander, J. : Towards Federated Learning at Scale: System Design. CoRR **abs/1902.01046**, (2019). http://arxiv.org/abs/1902.01046
7. Duan, M. : Astraea: Self-balancing Federated Learning for Improving Classification Accuracy of Mobile Deep Learning Applications. CoRR **abs/1907.01132**, (2019). http://arxiv.org/abs/1907.01132
8. Yao, X., Huang, T., Wu, C., Zhang, R., Sun, L. : Federated Learning with Additional Mechanisms on Clients to Reduce Communication Costs. arXiv e-prints, (2019). http://arxiv.org/abs/1908.05891
9. LeCunn, Y., Cortes, C., Burges, C. J. C. : THE MNIST DATABASE of handwritten digits. http://yann.lecun.com/exdb/mnist/
10. Federated Learning: The Future of Distributed Machine Learning, https://medium.com/syncedreview/federated-learning-the-future-of-distributed-machine-learning-eec95242d897
11. The New Dawn of AI: Federated Learning, https://towardsdatascience.com/the-new-dawn-of-ai-federated-learning-8ccd9ed7fc3a

12. Federated Learning: Collaborative Machine Learning without Centralized Training Data, https://ai.googleblog.com/2017/04/federated-learning-collaborative.html
13. Federated Learning: Collaborative Machine Learning without Centralized Training Data, https://venturebeat.com/2019/06/03/how-federated-learning-could-shape-the-future-of-ai-in-a-privacy-obsessed-world/