Mixed-mode Federated Learning Cloud-based System

Partha Pratim Saha [*] 2019AD04100@WILP.BITS-PILANI.AC.IN

Abstract

In Federated Learning (FL), the central server potentially represents a single point of failure - which is one of the bottlenecks of centralized federated learning. Another issue is the need for all participants to trust the central authority with their datasets. In contrast, a decentralized federated learning solution needs parties to run a common binary on each of their datasets and trust the incoming program. Another issue is the training run time due to multiple hops between different dataset locations. System designers often face a tradeoff between the higher performance of centralized federated learning vs. the better security of distributed federated learning. In this paper, we propose a novel Mixed-mode Federated Learning (MFL) solution that combines the advantages of centralized and decentralized federated schemes, without compromising security. The benefit of this proposal is to partition split data to be shared or kept private, to balance between performance and privacy considerations.

Keywords: Centralized and Decentralized Machine Learning, Deployment, Mixed-mode Federated Learning, Neural Network.

1. Introduction

Federated Learning is a machine learning technique where many clients (e.g., devices in an organization or across multiple organizations) collaboratively train a model. This training is done under the orchestration of a central server (e.g., provider) while keeping the training data decentralized. FL can mitigate many systemic privacy risks and costs by keeping private data on-site, any possibility of leakage during transmission and at a remote shared site is minimized, thus resulting from traditional centralized machine learning and data science approaches Mothukuri et al. (2021). The central server receives the dataset contributions from all clients. FL is typically used when one needs to train models on a larger dataset than any one single entity owns and is not willing to share its data with others (e.g., for legal, strategic, or economic reasons). FL trains an algorithm keeping the training data locally on users' decentralized systems rather than contributing it to a single data center for training. The distributed locations are used as nodes performing computation on their local datasets to update a global model Konecny et al. (2015). This contrasts with traditional centralized machine learning techniques, where all the local datasets are uploaded to a shared server location. If data types at different locations are different, then FL enables multiple participants to build a common, robust machine learning model without sharing their data, thus addressing critical issues such as data privacy, security, and access rights Xie et al. (2021) Bonawitz et al. (2019). Various applications for clinical and biomedical research are already exploring cross-device FL solutions Kairouz et al. (2021).

2. Cloud-Based Collaborative Tools

Imagine that multiple parties need to collaborate for a common purpose but do not trust each other with their data sharing. An example is research for a new drug that needs multiple hospitals to provide patient data, pharmaceuticals to provide their drug data, and medical researchers to explore new treatment protocols. Neither party may want to give away its dataset, but all are interested to know new drug protocols that are effective in treating a disease. In this situation, a multi-party cloud is a feasible solution Müller et al. (2016). In such a setting, multiple participants collaborate using shared tools. This requires data of each party to be kept private from other users while sharing the computed models for all to use.

A proposed framework for Secure Multi-Party Computation (SMPC) with four entities: proxy server, cloud server, analyzer, and parties are taking

[*] This work has been done as part of my masters dissertation at the Birla Institute of Technology and Science(BITS) Pilani, India

part in the shared computations Pussewalage et al. (2016). A proxy server hides the identities of all users to provide anonymity Kang et al. (2013). Each user can send its data for computations after authenticating it into the system. There are obvious questions on the performance and efficiency of cloud environments to deploy such a model while enforcing the security requirements of all user entities. We propose a sophisticated Mixed-mode FL (MFL) solution that enables knowledge transfer among parties (clients). Our approach maintains private data locally and improves communication efficiency, as shown in Figure 1. FL

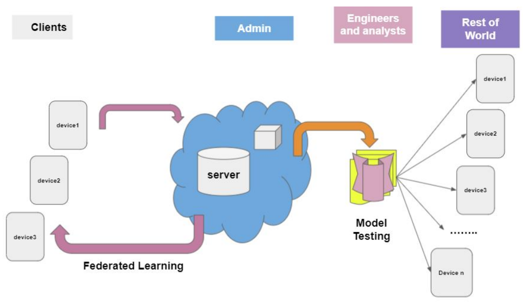

Figure 1: Collaboration among clients in FL setting Kairouz et al. (2021)

enables data to stay local, while algorithms travel across the participating institutions, to train the algorithm. This solution preserves the privacy and security of users' data. Consider three hospitals, namely Hospital A, Hospital B, and Hospital C, with a single centralized server. Some notations are below:
t_{da} = Data copying delays from Hospital A to the central server
t_{db} = Data copying delays from Hospital B to the central server
t_{dc} = Data copying delays from Hospital C to the central server
t_{pa} = Time for code and weights of Neural network to travel from the central server to hospital A
t_{pb} = Time for code and weights of Neural network to travel from the central server to hospital B
t_{pc} = Time for code and weights of Neural network to travel from the central server to hospital C
t_{px} = Program execution time
n = number of training iterations.
So, the total data copy time to central database is:

$$t_{da} + t_{db} + t_{dc}$$

and in a completely centralized model, total run time will be:

$$T1 = t_{da} + t_{db} + t_{dc} + n * t_{px} \tag{1}$$

For a fully decentralized Federated Learning system, total run time will be:

$$T2 = n * (t_{pa} + t_{pb} + t_{pc} + t_{px}) \tag{2}$$

For larger values of n, $T2 >> T1$, because, in the first model, we copy data only once, whereas, in the second model, the program has to travel once for each iteration. Even in a single iteration case, if $t_{pa} > t_{da}$, $t_{pb} > t_{db}$ and $t_{pc} > t_{dc}$ which will happen for a large program binary, $T2 > T1$. Even for smaller programs and larger dataset, since model training is done iteratively, n maybe ten or higher usually causing $T2 > T1$, even if the program binary is only $\frac{1}{10}$ th size of the dataset. However, in the cases when very few iterations are required for the training program, then $T1 > T2$ as shown in Figure 2. Below scenarios will depict these differences:

2.0.1. **Example** 1 **scenario:**

In this example, if the time to copy data from hospital A to the central server is 5 units, for Hospital B is 10 units, for Hospital C is 15 units, the run time for program P is 2 units, then to run the program for n iterations, it will take n*2. Furthermore, if the time for code and weights of Neural Network to travel to Hospital A is 1, Hospital B is 2 and Hospital C is 3 units, respectively.

We plot the number of iterations(n) and run time in a graph where the program size is smaller than the dataset size, as shown in Figure 2.

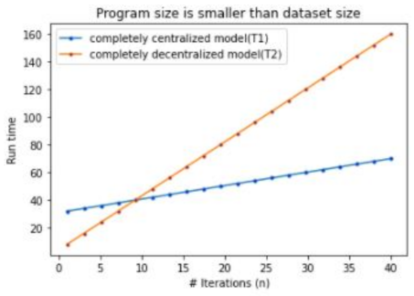

Figure 2: Program size is smaller than dataset size

Inference: In this scenario, initially the centralized model takes more time because more data is copied to a central server. However, after the tenth iteration the decentralized model becomes slower.

2.0.2. **Example 2 scenario:**

In this example, say if the time to copy data from hospital A to the central server is 1 unit, for Hospital B is 2 units, and for Hospital C is 3 units. the run time of program P be 2 units, then to run the program for n iterations, it will take n*2. If the time for code and weights of Neural Network to travel to Hospital A is 5, to Hospital B is 10 and Hospital C is 15 units, respectively.

We plot the number of iterations(n) and run time in a graph where program size is bigger than the dataset size, as shown in Figure 3.

Inference: In this scenario, a decentralized model is always slower because we do need to copy the program for every iteration.

In general, a decentralized model is slower most of the time because code and model need to travel to different locations. However, it is often preferred for security and privacy reasons. The implementation using python can be found here.

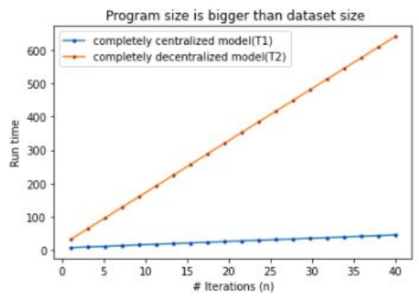

Figure 3: Program size is bigger than dataset size

Since Federated Learning can be implemented where shared data is collected centrally and analyzed in the central location, or data can remain distributed, the analysis (i.e., shared code and neural network weights) moves from location to location. In the first case, performance is primarily limited by the one-time movement of the data and the subsequent multiple iterations done centrally. In the second case, performance is limited by the multiple movements of the analysis mechanism (i.e., shared code and neural network weights), done once for each iteration, and the calculation is done in a distributed manner in Figure 4 Liu et al. (2019) Brendan McMahan et al. (2017). As the previous examples showed, at some point, performance of the centralized implementation outperforms the distributed case. The downside is the security and privacy issues with central data collection.

3. Mixed-mode Federated Learning

Mixed-mode Federated Learning (MFL) is a combination of centralized and decentralized machine learning algorithms. Each client has data of a different set of subjects, while data of every client has the same set of features. Examples of such data include mobile users' word-typing histories (from the same word corpus), which are stored on individual devices with the same dataset features and analyzed by private machine learning algorithms.

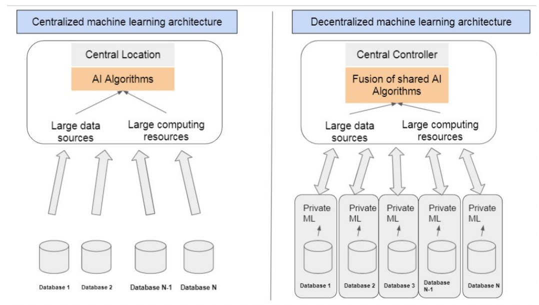

Figure 4: Centralized and Decentralized Learning

In Centralized Learning (CL), all the data is aggregated from multiple sources in a central location, where FL algorithms do the training and inference computations. Then results are distributed back to different clients. Each client can see only its own data and the final results of centralized computations, but not other clients' datasets. Hence data and models are shared.

In Distributed Learning (DL), each client keeps its data private, whereas the FL algorithm travels to each client's site, does some computations and partial results are then copied back to a central location. From there, FL algorithms go to the next client's site, do more computations, and update the central database. A client cannot see the data of others but only accesses the shared FL model. A key difference is that if clients do not trust each other, they never have to give their private data away. In this case, data will not get shared, only FL code is shared.

We propose dividing the data in two parts: private and public, as shown in Figure 5. Private data is similar to a patient's identifiable information in the

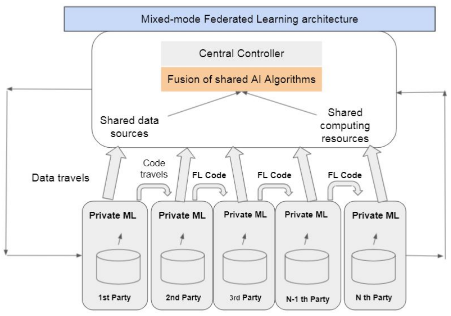

Figure 5: Mixed-mode FL architecture

healthcare domain. This information can be removed and replaced by an ID, which is only known to the owning party. We reckon that 90 percent of data can be in the other public participants so that it can be safely shared. For a non-healthcare example, people share their work profiles, designations on social networking platforms (e.g., LinkedIn) without sharing their Social Security Number (SSN) or personal salary information. This enables some meaningful computations to be done on a shared basis without having sensitive code travel between the sites resulting in unacceptable delays. Then computation results can be shared between the contributing parties, which may be able to do reverse mapping of public IDs to private information.

4. Challenges in Mixed-mode Federated Learning

1. Local and global models: In MFL training, each client has its local data. At the time of inference, however, a piece of input data may have limited or complete features. In the first case, the client can make an inference by itself. In the latter case, the client will rely on the central server to make the inference. So a server is required to maintain the model that supports all the features.
2. Limited data sharing: In typical MFL, clients do not share their local data or labels. This solution needs to deal with both types of clients, so it is expected that the training method can run without the need for the server to access any data, including the labels.
3. Sample synchronization: A typical issue with DL (where each client has some features of all training data) is that all the clients need to draw the same mini-batch of training data; this problem is exacerbated in the MFL system because not all the parties have all the samples. An ideal algorithm shall work without requiring the clients to synchronize their sample draws.

An unbalanced and non-IID (identical and independently distributed) data partitioning across a massive number of unreliable devices with limited communication bandwidth is also an additional challenge Li et al. (2021). It is clear that no existing FL methods can meet all these requirements better than our proposed MFL solution.

5. Potential applications of Mixed-mode Federated Learning

Applications of FL such as adapting pedestrians' behavior in traffic, cardio health events from wearable devices. We discuss a couple of applications below:
1. Learning over smartphones: By jointly learning user behavior across a large pool of smartphones to power next-word prediction, face detection, and voice recognition tasks. However, some users may not wish to transfer their smartphones's data to a central server due to privacy, bandwidth, or battery power concerns. FL can support predictive features on smartphones without compromising privacy or negatively impacting the user's experience. An application to perform next-word's prediction in a mobile network is based on text data in users' history Li et al. (2019). In this model, smartphones learn and update a global model from a central server.
2. Healthcare solutions: This may represent the final frontier for humanity to win over new and existing diseases that so far have not been conquered using the last 100+ years of technologies, as shown by stark similarities between the 1918 and 2019 pandemics. This requires cloud based ML solutions Gupta and Sehgal (2021). Enormous other possibilities exist using cloud-based data repositories and ML tools Sehgal et al. (2020) Gupta and Sehgal (2021).

6. Conclusion

This paper proposes a solution architecture that combines the advantages of centralized and decentralized federated learning techniques. In a centralized system, all participants are expected to contribute their datasets in advance, which is not always feasible given various privacy issues and security concerns. In a decentralized system, each participant retains its dataset while the training algorithm travels between the sites. We described the security risks and potential performance issues. We propose to overcome both the security and performance issues with a new Mixed-mode Cloud based System. A model has been built to show the decision points when program size is bigger or smaller than the dataset sizes with proposed Mixed-mode Federated Learning system, and enumerate its challenges. We conclude the paper with a brief description of potential applications. Our proposed architecture has enormous potential to address the global healthcare data sharing issues, as exposed by the recent pandemic.

References

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecn, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design, 2019.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Agüera y Arcas Blaise. Communication-efficient learning of deep networks from decentralized data, 2017.

Pramod Gupta and Naresh K. Sehgal. Introduction to machine learning in the cloud with python, 2021.

Peter Kairouz, H. Brendan McMahan, and Brendan Avent et. al. Advances and open problems in federated learning, 2021.

Ruogu Kang, Stephanie Brown, and Sara Kiesler. Why do people seek anonymity on the internet? informing policy and design. Technical report, Carnegie Mellon University, 2013.

Jakub Konecny, H. Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter, 2015.

Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He1. A survey on federated learning systems: Vision, hype and reality for data privacy and protection, 2021.

Tian Li, Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. Federated learning:challenges, methods, and future directions. Technical report, Carnegie Mellon University, Bosch Center for Artificial Intelligence, 2019.

Yiming Liu, F. Richard Yu, Xi Li, and Hong Ji. Illustration of different machine learning architectures. Technical report, Beijing University of Posts and Telecommunications, 2019.

Viraaji Mothukuri, Reza M.Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning, 2021.

Henning Müller, Jayashree Kalpathy–Cramer, Allan Hanbury, Keyvan Farahani, Rinat Sergeev, Jin H. Paik, Arno Klein, Antonio Criminisi, Andrew Trister, Thea Norman, David Kennedy, Ganapati Srinivasa, Artem Mamonov, and Nina Preuss. Report on the cloud-based evaluation approaches, 2016.

Harsha S. Gardiyawasam Pussewalage, Pasika S. Ranaweera, Vladimir A. Oleshchuk, and & BIndika A. M. Balapuwaduge. Secure multi-party based cloud computing framework for statistical data analysis of encrypted data, 2016.

Naresh K. Sehgal, Pramod Chandra P. Bhatt, and John M. Acken. Cloud computing with security. Technical report, Intel Corporation USA, IIT Delhi, Portland State University, 2020.

Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang. Multi-center federated learning, 2021.