Spring 5-7-2021

Decentralized Aggregation Design and Study of Federated Learning

Venkata Naga Surya Sameeraja Malladi

DECENTRALIZED AGGREGATION DESIGN AND STUDY OF FEDERATED LEARNING

A Thesis Presented to
The Faculty of the Software Engineering

by

Venkata Naga Surya Sameeraja Malladi

In Partial Fulfillment
of Requirements for the Degree
Master of Science, Software Engineering

Kennesaw State University
May, 2021

In presenting this thesis as a partial fulfillment of the requirements for an advanced degree from Kennesaw State University, I agree that the university library shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish, this thesis may be granted by the professor under whose direction it was written, or, in his absence, by the dean of the appropriate school when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from or publication of, this thesis which involves potential financial gain will not be allowed without written permission.

_____

Venkata Naga Surya Sameeraja Malladi

ABSTRACT

The advent of machine learning techniques has given rise to modern devices with built-in models for decision making and providing rich content to users. This typically involves processing huge volumes of data in central servers and sending updated models to end-user devices. There are two main concerns on this server architecture, one is the privacy of data that is being transferred to a central server and the other is volumes of data sent over the network for the model update. Federated Learning helps solve these problems by training models on local data within the device and aggregating the model with other devices. Federated Learning involves a central server for the aggregation and the resulting updates to the clients, here only model parameters are shared with the central server not the data itself thereby preserving privacy. But all the applications are not being compatible with Federated Learning and also there is a privacy concern of models being shared to the central server which can be susceptible to malicious attacks. In this paper, central server free Federated Learning, which is decentralized Federated Learning is used, where the parameters will be exchanged between the clients one to one and get their models updated removing the need for a central server for aggregation. Peer-to-peer techniques are used for communicating between clients and different node architectures to achieve better accuracy. This happens when the clients meet another client in a connected social network environment. The results show that the communication happens between clients in a decentralized fashion and thereby achieving privacy in a more trusted manner.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Machine Learning advancements are bringing continuous changes for the needs and ease of many products and services. The introduction of smart devices such as Internet of Things (IoT) devices [Ahamed and Farid, 2018], where the user data is collected directly and stored in some third-party cloud systems, is now ubiquitous in all sectors. The smart devices that collect personal information regarding health, finances, and business might get compromised by the vulnerabilities of these IoT and other smart devices. Attackers stealing sensitive information from these IoT devices[Pang et al., 2021] can cause severe impacts for the affiliated companies. This kind of data intrusion happens as the users raw data is directly stored and computed in the servers. A deep learning framework named "Federated Learning" [Yang et al., 2019] addresses this privacy concern in a secured fashion. Data training occurs now on the user's devices in the Federated approach, and just the parameters will be stored and processed on a central server.

Federated computation falls under the category of edge computation. The data training happens on the edge devices. Federated Learning is a framework that establishes safety with the help of an aggregation secured by gathering data from a massive volumes of users. This technique eliminates the possibility of significant

security threats of consumption of the user's original data. Federated Learning aggregates the model parameters in a central server and sends them back to the users for a better model, thereby improving the model's performance. Despite being named a secure model, the Federated Learning framework also has some privacy limitations. A single server cannot guarantee the security standards. The hackers can also make reverse-engineering attacks, get the clients information, and even hack the aggregated algorithm sent back to the clients, the targeted and improved version. Fig. 1.1 represents the architecture of Federated Learning.

Despite its secure aggregation [Bonawitz et al., 2016], Federated Learning has its limitations in the form of communication cost too. Apart from this, having a single server is not promising in the case of data security. The server is prone to undergo Sybil attacks [Douceur, 2002] where the hackers introduce fake clients by manipulating the central server and send the parameters so that the quality of the final model degrades. This poisoning attacks [Cao et al., 2019] change the label of the parameters that need to be aggregated with a look-alike dataset. Also, the attackers create a GAN network [Zhang et al., 2019], unnoticed by the server, introduce false labels and thereby affecting the local update, which gets worse after multiple epochs, achieve their goal of degrading the model accuracy.

Keeping all these concerns on a list, I designed a Decentralized Federated Learning framework where there is no scope for central server-based attacks as there will be no central management system/central server. The user's parameters are shared among the peers securely. This process happens by utilizing the "Pysyft" framework, an open-source python library built on top of "Pytorch," another python library where both work for the simulation and validation of the proposed "Decentralized Federated Learning." In this Decentralized Federated Learning, the users exchange their parameters in different architectures and based on the better convergence results for the model. Fig. 2.1 represents the architecture of Decentralized Federated Learning.

In the proposed methodology, I have 1) Designed a framework "Decentralized Federated Learning". 2) Introducing ring and mesh architectures for the aggregation scenario corresponds to the Federated Learning aggregation scenario. 3) This, in turn, happens by adding a new method/process to the "Pysyft" to validate this work. In this paper, I will also discuss the different architectures that are applied while aggregating the parameters and discuss which is a better choice in the perspective of convergence of the results. I have aimed for better accuracy while the users exchange their parameters in central server-free computations. In this work, I proposed a parameter exchange algorithm where the pointer tensors from pysyft library played a key role in achieving the main objective.

The contribution of this paper can be highlighted as follows.

- Introduction of new methodology to existing Pysyft framework.
- Designing new features through different topologies.
- Impact of the factors that played a crucial role in the betterment of accuracies.

This paper is organized as the following. Chapter 2 shows the existing Federated Learning, the drawbacks of it and the related contributions. Chapter 3 depicts the system implementation of the proposed "Decentralized Federated Learning". Chapter 4 provides the simulations and Results. Chapter 5 summarizes the conclusion and future directions of this Decentralized Federated Learning.

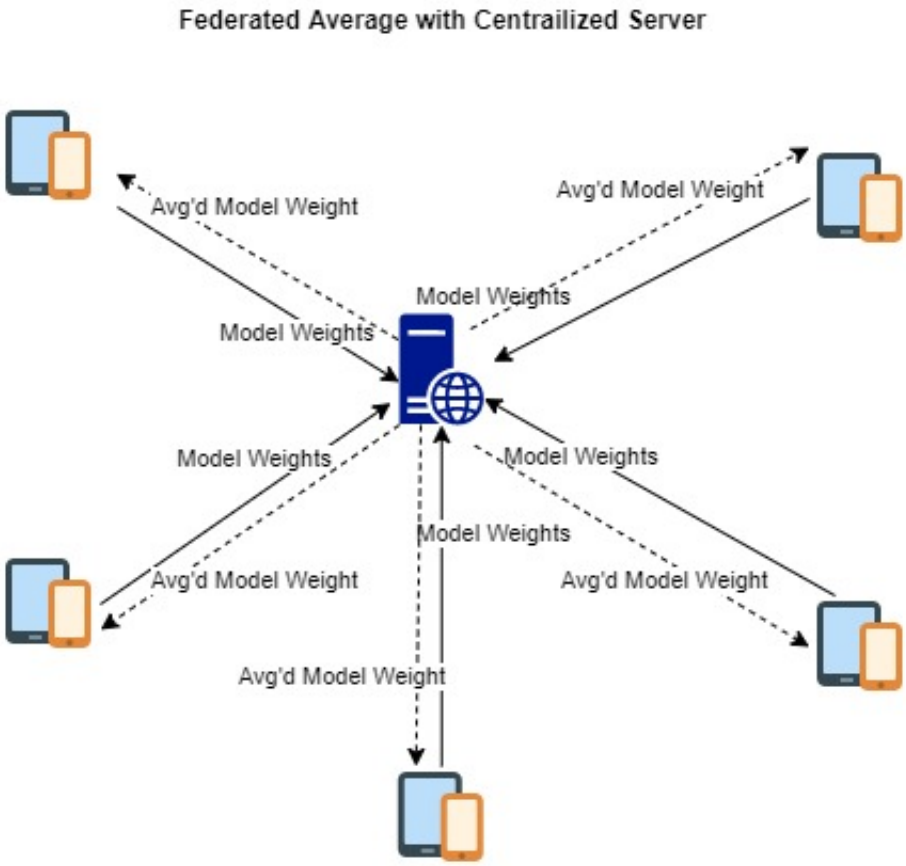

Figure 1.1: Conventional Federated Learning Structure

Chapter 2

Related Work

In this chapter, the existing related works can be seen on Federated and Decentralized Federated Learning and compare with our contributions.

2.1 Federated Learning

Federated Learning is a framework, which secures the massive data which is highly decentralized, by getting trained at the local server. The training happens in multiple rounds,until the model gets good convergence result as per the standards set by a data scientist. The users are treated as virtual workers and the parameters are shared with the central management system where the actual aggregation happens with Differential Privacy (SMC) [Wei et al., 2020] and Secure Multiparty Computations [Truex et al., 2019]. The parameters are aggregated in each round of training. The updated model now is sent to the worker, here the worker can be a device, the training now continues on that updated model and the new parameters are sent again to the central server for second round of aggregation. This continues until the desired model is obtained.

There are some techniques that reinforce the security standards of FL framework. Firstly, differential privacy is where artificial noise is added at the client side and

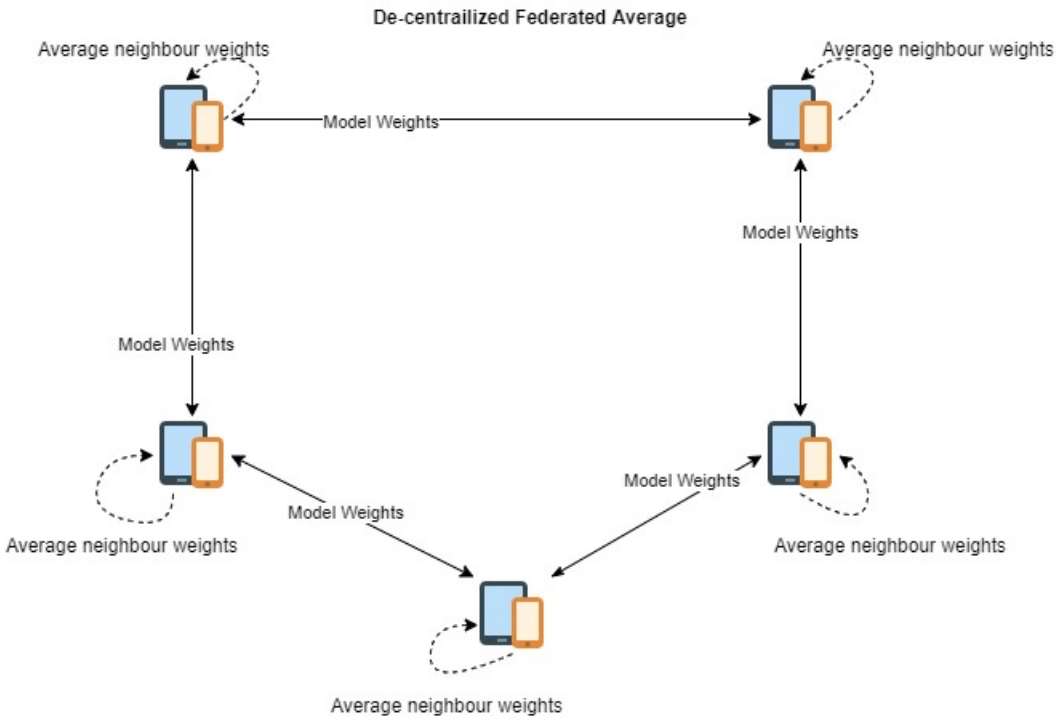

Figure 2.1: Decentralized Federated Learning Structure

the server side to the model parameters that have to be aggregated. This masks the parameters from the perspective of data intruders. The noise can be scaled. But the downside of that can be privacy loss in the name of masking it through adding noise. Secondly, FL utilizes a technique named Homomorphic Encryption, where the secret key is shared additive among the participants in the FL framework and gets decrypted with only some threshold value. Sometimes, the noise added by the differential privacy can account for the poor performance at the end of aggregation. So to address this issue, the technique called Secure Multiparty Computation is deployed in Federated systems.

In the Federated environment, the results are more converged when there are more clients participating in the aggregation which is Federated Averaging [McMahan et al., 2017], in technical terms. The performance of the model is gauged in the Federated framework is in the form of privacy protection and communication costs. In this work, I focused mainly on the privacy protection but not much on

the communication costs. I have also considered the time taken for the convergence and the collaborative learning for the system. I have also concerned the Federated Learning work by observing all the necessary changes happen at all the stages of this learning process.

2.2 Drawbacks of Federated Learning

Although, Federated approach addresses the data security by training at the edges, the issues of data privacy is still existing in the form of poisoning attacks. Apart from these external and internal privacy attacks, the conventional Federated Learning encounters the heterogeneity issue [Pang et al., 2021] when the data is more skewed and the Non-IID data. Due to this heterogeneity, the quality of the ML model will be dwindling. Keeping all this concerns, I proposed a novel framework "Decentralized Federated Learning" which will address the strong motive,data privacy and security by a good convergence and accuracy of the trained model.

2.3 Decentralized Federated Learning

There are many works which discusses on fully Decentralized Federated Learning. Majority of works uses peer-to-peer techniques to achieve Decentralized Learning. Paper [Lian et al., 2017] discuss the Decentralized Federated Learning using Decentralized Parallel stochastic gradient Descent technique. Here the models are trained parallel in each local worker and are aggregated by sending the trained model parameters to its neighbour nodes. My work is similar to this approach except that I do not aggregate all worker parameters at once, instead I selected pair of workers at a time and aggregate them and continue until all pairs are considered. My major contribution and focus is on achieving the better accuracy of the Machine Learning model in particular iterations of the model that I have trained. Also, their objective is to

prove the rate of communication with respect to the computational complexity while my work is based on improving the accuracy of the model with not more than 500 epochs with a good convergence results in terms of high accuracy and lower loss. The paper [Hu et al., 2019] discussed about the network bandwidth between the nodes and the network congestion based on a segmented gossip aggregation protocol.

Chapter 3

System Implementation

This chapter shows the different versions and topologies in which a given machine learning model is considered, trained, and tested for its accuracy and loss with the Decentralized Federated Learning as the base architecture. I will also discuss the corresponding algorithms of the major topologies that I have contributed to ther experiment's main objective.

3.1 Introduction of new methodology to existing Pysyft framework

In this approach, the users are the mobile or edge devices which include this IoT device that collects the data from the device through the user's actions and inputs. I have designed the system concurrently with three methodologies with the major focus on achieving better accuracy. Those three different machine learning algorithms are: 1) Individual Remote training 2) Conventional Federated Learning 3) Decentralized Federated Learning To achieve this, I have developed the Decentralized scenario with the help of "Pysyft" framework, where I have created a method that exchanged the model parameters between two workers. Pysyft is built on top of other python library

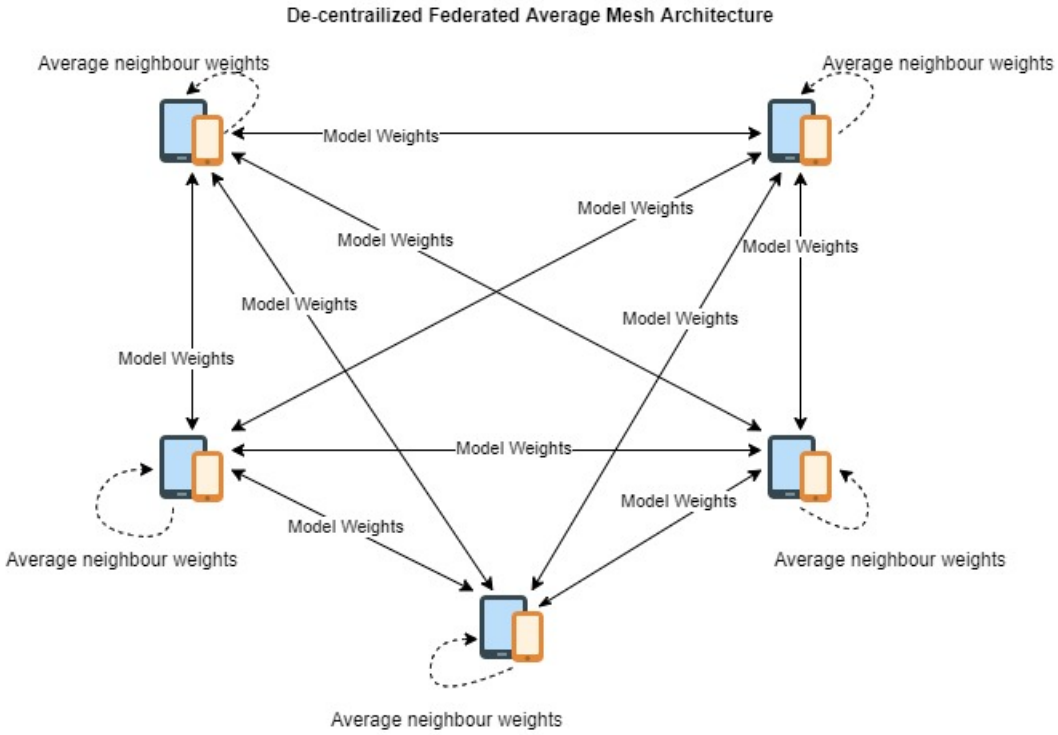

Figure 3.1: Mesh architecture for the Decentralized framework

called "Pytorch". Pytorch is predominantly used in deep learning frameworks and technologies. The principal factor of exchanging the parameters between the workers in the decentralized fashion is achieved by the concept of moving pointer tensors between the workers. The pointers will be stored in this process. This feature sums until achieving the main objective of successful aggregation after exchange. Here, the model will be trained on the devices (say two workers), then they will be aggregated on either side of the workers. To make this exchange happen, then events will be created with events table. In each event, any two workers exchange their local parameters. Here, in this initial scenario, I considered a total of three workers, and they all exchange the parameters. This is observed under homogeneous distribution. In this model, the accuracy results of both the centralized and decentralized frameworks are in a similar fashion, but they are not a good measure. Later the scenario is developed with a difference in the data distribution, which is now heterogeneous of its kind. The key role is played by the number of workers that participated in each epoch of

training and aggregation.

3.2 Designing new features through different topologies

The Decentralized Federated Learning is achieved in three main topologies in our work, where two workers exchange their parameters at any given time, and that happens according to the events in the events table. The training will be happening, and later, the workers will exchange the parameters, and this happens with respect to the homogeneous and heterogeneous data distribution wherein one can understand the fundamental differences in achieving the accuracy results in both the models that come under this flat architectural style, which will not give us the expected accuracy results. This leads us to validate our work with the help of other architectural styles such as Mesh and Ring architectures, where the network will be fully connected, and all the workers exchange their parameters in pairs, before which the training of the desired model takes place. Leaving a certain worker at a particular iteration makes the jumpy accuracy of the overall collaborative learning at that particular round of communication. As the aggregation score drops down sharply because the worker not participating in the communication makes the accuracy fall below the average accuracy and drops completely. The workers aggregate their parameters on each of them. This happens until the desired accuracy results are obtained. In the ring architecture, it is fully connected in a kind of ring architecture in the communication topology where the worker 1 will be connected to worker 2, and it connects in the same fashion until the last node or last user in the network connects back to the worker 1. Here in this fashion, there will not be any network congestion issues. This reduces the communication overheads. But the timing factor plays a quiet role, where the worker's unavailability during the communication or aggregation might be challenging. But

that will not sum up to bigger proportions as there will be workers available at any given time because of the ubiquitous mobile devices or edge devices, which are willing to participate in the communication and learning. The other architecture proposed is the mesh architecture, where the nodes/workers are connected in a way where every worker has connected all other workers in a mesh topology or star topology as per the computer networking. Mesh architecture in the decentralized environment is depicted in Fig. 3.1. Here, the network is better utilized, but traffic congestion might play a greater role in the trade-off for achieving better accuracy results.

3.3 Impact of the factors that played a crucial role in the betterment of accuracies

The factors that impact the accuracy of the Decentralized Machine Learning are the architectural patterns that are built with the help of "Pysyft" and "Pytorch" with the usage of the exchange multiple parameters algorithm that led to better accuracy on the iteration. This methodology exhibits lower privacy loss and higher accuracy with the ring architecture overall. The major setback is solved by including all the workers during the learning process. The learning process includes both communication and aggregation. The cause for the good results is not excluding any of the participants during or before the learning process. In our work, the Pysyft features aided in building the method that takes the parameters from the participants and where it involves no central aggregation or storage of weights of the neural network. The process perhaps helped in aggregating the parameters on the side of the participants/devices.

This concurrent comparison shows the promising results for our proposed parameter exchange algorithm for the Decentralized Federated Learning algorithm. The individual remote training is nothing but each worker trains on their local data. Here no exchange of data or parameter takes place. Then comes the traditional Federated

Learning algorithm that uses the "Federated Averaging". Here, the on-device training of the machine learning model happens, and the individual parameters from massive users will be sent to the central server where the aggregation of the parameters happens, and the new updated model will be sent to all those devices, and this process continues till the organization or the data scientist gets the desired model. Finally, the decentralized approach, which is our main focus, wherein there is no central server that receives these aggregated parameters. The decentralized kind of aggregation happens amongst the users rather than the central server/parameter-server.

The algorithms for all three different approaches for Decentralized Federated Learning are depicted and explained in the below algorithms.

Algorithm 1 Decentralized algorithm for three workers exchange topology

Input: $dataset$,$params$,$worker_1$,$worker_2$,$worker_3$ $model$

Output: $(accuracy, loss)$

1: *initialize the workers*
2: **for** $i \leftarrow 1$ **to** n **do**
3: *train the workers with model in mini batches*
4: $Exchangetheparameters\ params1$,$params2$,$params3$
5: $(params_{updated1} \leftarrow params_1 + params_2)/2$
6: $(params_{updated2} \leftarrow params_2 + params_3)/2$
7: $(params_{updated3} \leftarrow params_1 + params_3)/2$
8: $(params_{aggregated} \leftarrow params_{updated1} + params_{updated2} + params_{updated3})/3$
9: *test the model*
10: **end for**

In the Algorithm 1 with the basic intent of testing the features of Decentralized Federated Learning algorithm, I have considered it for three workers in a round of decentralized training. In this algorithm, I took the data set, the three workers required for the exchange, and the model for training. Expected is the output in the measures of accuracy and loss after corresponding testing and training of the model. In line 1 in the Algorithm 1, the workers from the events table are initialized. In line 2, the control will enter the iterations or rounds of training for decentralized communication. In line 3, the loop for training each worker starts. In line 4, the

model parameters are exchanged between the workers. In line 5, the parameters are averaged between worker 1 and worker 2. In line 6, the parameters are averaged between worker 2 and worker 3. In line 7, the parameters are averaged between worker 3 and worker 1. In line 8, all the three individually exchanged, aggregated parameters are total aggregated and averaged to get the better performance. In line 9, the model will be tested and the accuracy along with loss measures are obtained.

Now the algorithm for mesh topology is depicted and explained below.

Algorithm 2 Decentralized algorithm for mesh topology

Input: $dataset$,$params$,$worker_1$,$worker_2$,$worker_3$ $model$
Output: $(accuracy, loss)$

1: *initialize the workers*
2: **while** $True$ **do**
3: **for** $worker_i \leftarrow 1$ **to** n **do**
4: *train the workers with model in mini batches*
5: $Exchangetheparameters\ params1$,$params2$,$params3$
6: $(params_{updated1} \leftarrow params_1 + params_2)/2$
7: $(params_{updated2} \leftarrow params_2 + params_3)/2$
8: $(params_{updated3} \leftarrow params_1 + params_3)/2$
9: $(params_{updated3} \leftarrow params_{n-1} + params_n)/2$
10: $(params_{updated3} \leftarrow params_2 + params_n)/2$
11: .....
12: $(params_{aggregated} \leftarrow params_{updated1} + params_{updated2} + ... + params_{updatedn}/\mathrm{n}$
13: *test the model*
14: **end for**
15: $updatedaccuracy ==$ *the desired accuracy*
16: **end while**

In the Algorithm 2, the model trained algorithm will be having the same inputs given to the previous algorithm, which are the parameters of the workers model, workers, data set. The output is measured in terms of accuracy and loss. In line 1, the workers from the events table are initialized. In line 2, the control will enter the iterations or rounds of training for decentralized communication. In line 3, the loop for training each worker starts. In line 4, the workers are trained with the desired model. In line 5, the users/participants exchange the parameters in such a way they were divided into equal mini-batches as per the training data set. In line

6, the parameters are exchanged and aggregated between any workers participating in the training. This process continues till line 11. In line 12, all the parameters are aggregated, finally the same as in the Federated Learning scenario. In line 13, the model will be tested. In line 15, the results will be compared with the desired accuracy levels.

Now, the algorithm for ring topology is depicted and explained below.

Algorithm 3 Decentralized algorithm for Ring topology

Input: $dataset$,$params$,$worker_1$,$worker_2$,$worker_3$ $model$
Output: $(accuracy, loss)$

1: *initialize the workers*
2: **while** $True$ **do**
3: **for** $worker_i \leftarrow 1$ **to** n **do**
4: *train the workers with model in mini batches*
5: $Exchangetheparameters\ params1$,$params2$,$params3$
6: $(params_{updated1} \leftarrow params_1 + params_2)/2$
7: $(params_{updated2} \leftarrow params_2 + params_3)/2$
8: $(params_{updated3} \leftarrow params_1 + params_3)/2$
9: $(params_{updated3} \leftarrow params_{n-1} + params_n)/2$
10: $(params_{updated3} \leftarrow params_n + params_1)/2$
11: ....
12: $(params_{aggregated} \leftarrow params_{updated1} + params_{updated2} + ... + params_{updatedn}/\mathrm{n}$
13: *test the model*
14: $sum\ \mathrm{n(n+1)/2}$
15: $accuracy = sum/(params_{aggregated}$
16: **end for**
17: $updatedaccuracy == the\ desired\ accuracy$
18: **end while**

In Algorithm 3, the algorithm is given the similar inputs as the previous two algorithms. In line1, the workers from the events table are initialized. In line 2, the control will enter the iterations or rounds of training for decentralized communication. In line 3, the loop for training each worker starts. In line 4, the workers divided into the mini-batches will be trained on the desired machine learning algorithm. In line 5, the parameters are exchanged between the workers. Here the exchange happens only with the neighboring nodes, unlike the mesh topology. This continues till line 11. In

line 12, all the parameters are then aggregated. In line 13, the model will be tested with the test data set. In line 14, the characteristic feature of ring topology are being observed, where the aggregated sum of a ring architecture expression is depicted in terms of "n" users. In line 15, the accuracy is checked for the desired accuracy.

Chapter 4

Simulations

In this chapter, I simulate the designed Decentralized Federated Learning to validate the performance and study the factors that can influence the performance.

4.1 Simulation Setting

These are done in different stages that improved the performance of the model by the increase in the rate of accuracy and lesser loss rate. The stages are described as the following:

- First stage is to develop a decentralized exchange parameter algorithm using Boston housing data.
- Second stage is to run the simulation with MNIST data set (with three virtual workers) using the exchange parameter algorithm with homogeneous data distribution.
- Third stage is similar to the second stage. I have used MNIST data set but with heterogeneous data distribution and exchange parameters between two workers at a time in each epoch.

- Next few stages are an attempt to resolve and understand accuracy jump in the third stage with heterogeneous data and exchange parameters between two workers at a time in each epoch.

- Next stage is a modified version wherein instead of exchanging between only two workers in each epoch, I have exchanged between two workers but involving all workers taking two at a time. Mesh topology is implemented here.

- In the next stage, I have considered a modified version of the earlier exchange parameter algorithm and mesh topology and run with 100 users.

- Data distribution for 100 users is that, assign three targets MNIST label to each worker. Results are not promising either due to less data being available to each worker or network lag.

- In the next stage, I considered only ten workers and run the mesh topology version. The results were good.

- In the next stage, I have used ring topology instead of mesh topology, i.e., I assumed workers are connected in ring fashion, and then exchange parameters between two consecutive workers in each epoch. It is observed that ring topology required more epochs to converge than mesh topology.

- Mesh topology accuracy converges after 150 epoch, whereas ring topology required 600 or more epochs for converging.

The development process, obstacles, and results are discussed below. I have used the "PySyft" library for our simulation and development. "PySyft" is an open-source library used for secured remote machine learning. Here, workers are mobile or edge devices where the data is generated and available locally. I have run three different learning methods in each simulation and compare the results. Data distribution for all three methods is the same.

First is an individual remote training. Here each worker trains on their local data. No exchange or average of parameter occurs. The second is a Federated Average method. Hereafter training individual worker models, model parameters are aggregated in the central server, which is the device on which simulation runs. The third is a D Decentralized algorithm which is our main objective of the proposed work.

There are two main data distribution methods that were used in the simulations on the MNIST dataset. The first one is homogeneous data distribution, wherein each worker gets all MNIST target values. Data is split into smaller mini-batches and is distributed uniformly across all workers. The second is heterogeneous data distribution. Here few MNIST target values (0 - 9 numbers) to each worker are assigned. Typically three numbers to each worker or overlapping numbers between workers. This is to simulate real-world examples wherein each local device may have data not seen on other devices, but the model should be able to work on all data. In heterogeneous data distributed, data is still split into mini-batches for each worker.

4.2 Homogeneous Distribution

The Algorithm 1 explained in Chapter 3 uses an events table wherein each row represents two worker Ids between which exchange of parameter occurs in each iteration. In this version, the exchange happens for each mini-batch iteration. This process assumes uniform data distribution across all workers, where each worker has the same number of mini-batches. The results in Fig. 4.1 and Fig. 4.2 show worker 1 and worker 1 result from three training results. In this result, both the centralized FL accuracy and Decentralized FL accuracy are the same. Then, I have tried with another version. The graphs for homogeneous distribution Fig. 4.1 representing the worker 0, and Fig. 4.2 representing worker 1 shows that the Federated Learning and the Decentralized Federated Learning are getting to converge at the end of the training

process. However, Decentralized Federated Learning is not at the same level as Federated Learning's accuracy levels. This training process is done with a limited number of workers and epochs less than 100. These results, though are not promising, lead us to the next model for experimenting with different data distributions and scaling the size of the training process by increasing the number of workers and number of iterations of the aggregation.

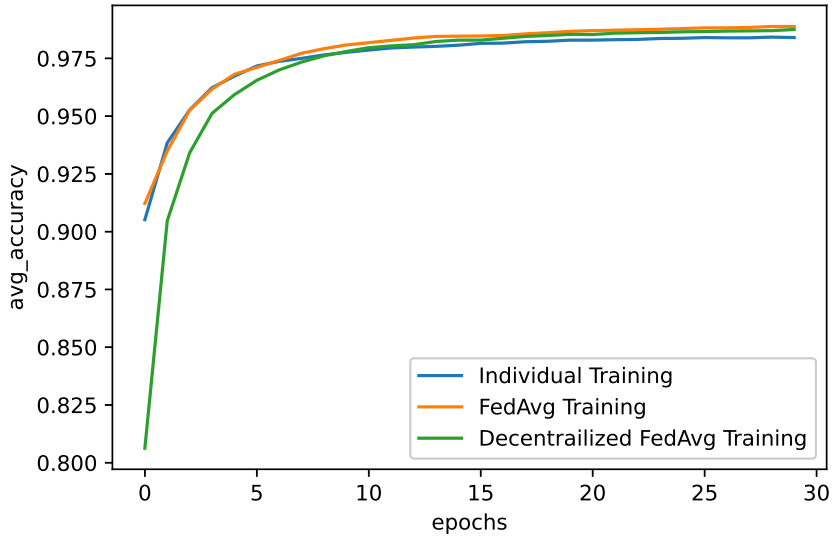

Figure 4.1: Homogeneous Simulation-a

4.3 Heterogeneous Data Distribution

Here, the exchange happens after complete training on all mini-batches in each epoch. This version does not assume the same number of mini-batches for all workers. Each worker can have any number of mini-batches since exchange happens only after complete training. Results show worker 0 and worker 1 results from three training results. As shown, the results are not as expected. Accuracy jumps in each epoch. The higher accuracy levels only when exchange occurs for pair of workers can be observed. We can see that pattern in Fig. 4.3 and Fig. 4.4. The individual training has poor per-

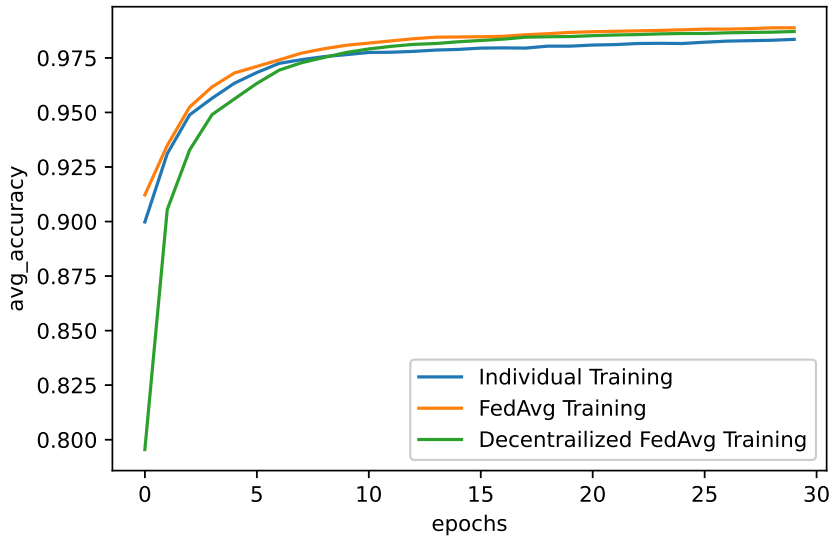

Figure 4.2: Homogeneous Simulation-b

formance in this case, and the accuracy seemingly has severe jumping frequency due to the exclusion of a worker. Here the Federated Learning has better performance in comparison. Then, I have performed another attempt to fix the accuracy jump that is seen above. Accuracy drops after training averaged model. This behavior is typical for the centralized Federated algorithm too.

4.4 Heterogeneous Roll back Architecture

To fix the accuracy jump, I have checked the accuracy after training and revert to the previous epoch where accuracy is higher. The results are shown in Fig. 4.5 and Fig. 4.6 representing worker 0 and worker 1 from three training processes. As you can see, accuracy becomes constant after some time since it is rolled back to the previous epoch, and there is no further movement. The rollback simulation graph Fig. 4.5 for worker 0 depicts that the accuracy levels of Decentralized Federated Learning is far below the accuracy level of Federated Learning as every time the lesser accuracy of the current iteration can be seen. Then it is rolled back such that accuracy to

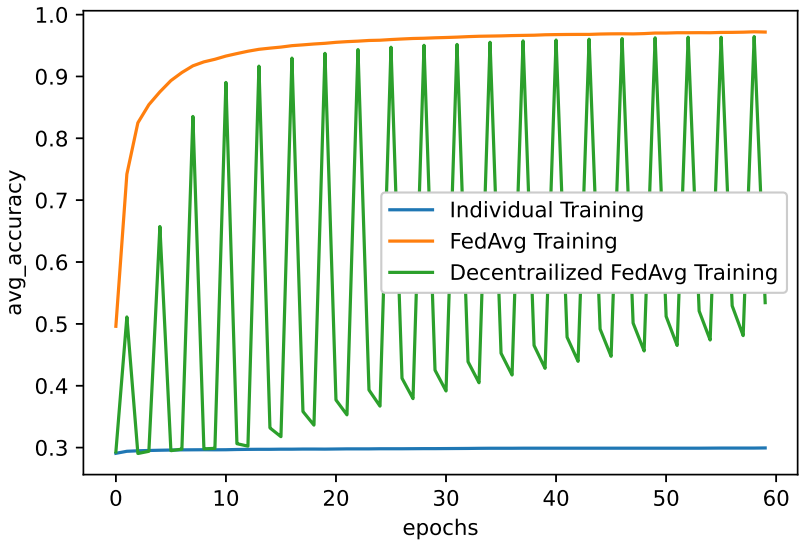

Figure 4.3: Heterogeneous Simulation-a

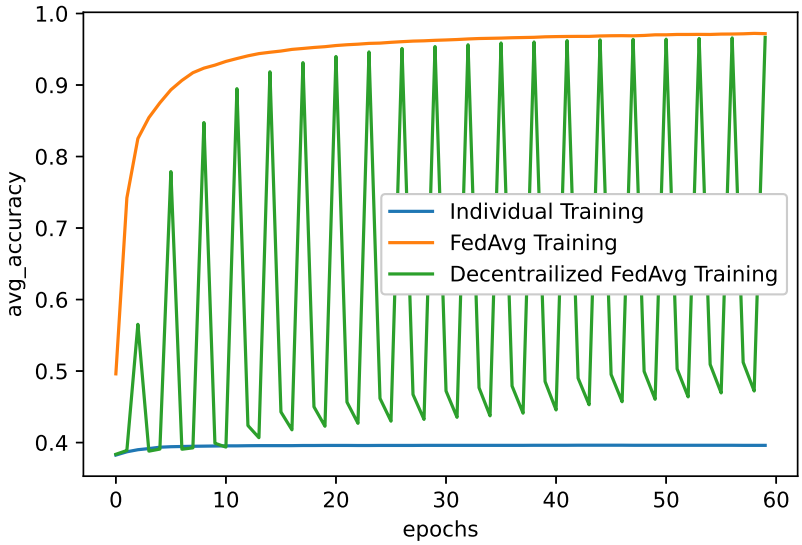

Figure 4.4: Heterogeneous Simulation-b

the previous accuracy levels. I performed this to avoid sharp drops in the accuracy levels. However, the performance is not as improved because the accuracy becomes saturated after certain rounds of training.

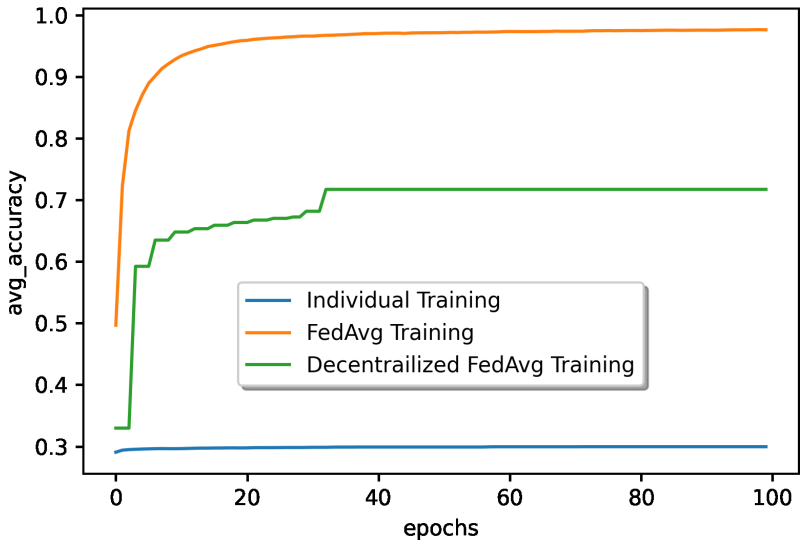

Figure 4.5: Roll back Simulation-a

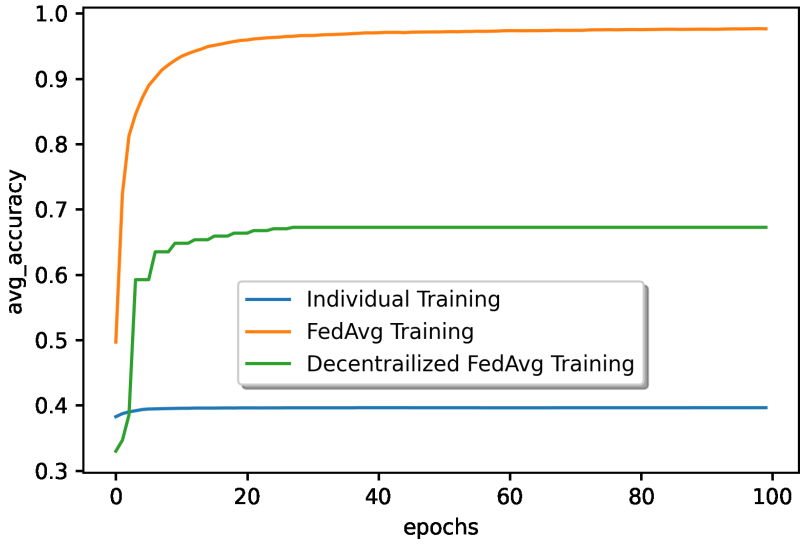

Figure 4.6: Roll back Simulation-b

4.5 Mesh Topology

This version is different from previous topologies. Here, the exchange of parameters between two workers takes place and ensured that all the workers are updated in each epoch. For example, three workers with worker id's 0, 1, 2 and 3. Then generated

exchange parameters list as: $[(0,1),(1,2),(0,2),(1,3),(0,3),(2,3)]$. Here no worker is excluded in any round of iteration. The performance of the training improves by such kind of organization of the nodes. Exchange happens between each pair of the list in each epoch. This approach is mesh topology where all nodes are fully connected and exchange parameters. After running with three workers, results were promising using a mesh topology architecture. I ran this simulation with 100 users, but the results are not as exact as that of Federated Learning. The accuracy has to be improved to a much more considerable extent. This dwindling accuracy can be due to the data distribution or network lag.

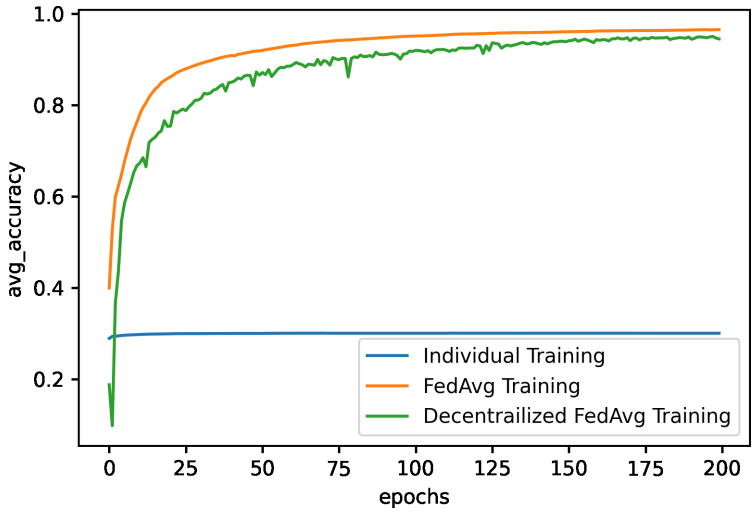

Figure 4.7: Mesh Topology Simulation-a

By taking limited number of users, this simulation achieved good results. Then, ran 200 epochs and seen they converge over the accuracy. This pattern can be depicted in the Fig. 4.7 and Fig. 4.8. This topology on a larger scale yields us the best results which mean that the accuracy levels are as good as that of a Federated Learning and the loss is low in this case. As no user is excluded in any round of training. This topology makes the accuracy levels converge on a good note at the increasing number of iterations. The major drawback can be the communication cost as all the nodes

and their combinations participate in the training process. We can observe the graph in the Fig. 4.7 where the Decentralized Federated Learning took a lot of time to reach a good level of accuracy as similar to that of Federated Learning. It is however, a good result after running multiple iterations. The individual training remained as poor as expected.

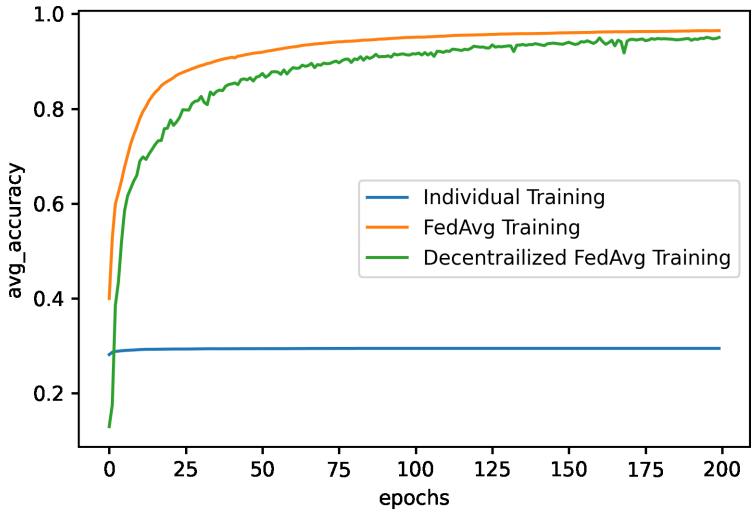

Figure 4.8: Mesh Topology Simulation-b

4.6 Ring Topology

This version is similar to mesh topology (version-3). The only difference is that nodes are not fully connected. In the ring topology, nodes are connected in a circle and communicate with their neighbors in each iteration. Only consecutive nodes exchange parameters pair-wise. The list of pairs for the sample 4 workers would be [(0, 1), (1, 2), (2, 3), (3, 0)]. Like Mesh Topology, I have simulated with ten users. It is observed that ring topology requires more converging iterations than mesh topology, which converged in 200 epochs. For ring topology, here I ran 600 epochs to see good results. The resulting accuracy graph can be depicted in Fig. 4.9 and Fig. 4.10. Here

in these figures, the accuracy levels of Decentralized, Federated Learning are so similar to that of the accuracy of a Federated model. The convergence levels are good with the increasing number of epochs. Here the major observation during the simulation is to get the results converged to better accuracy levels. This ring topology is taking longer times when compared to the mesh topology. But the levels will be very well converged with a huge number of users and iterations.

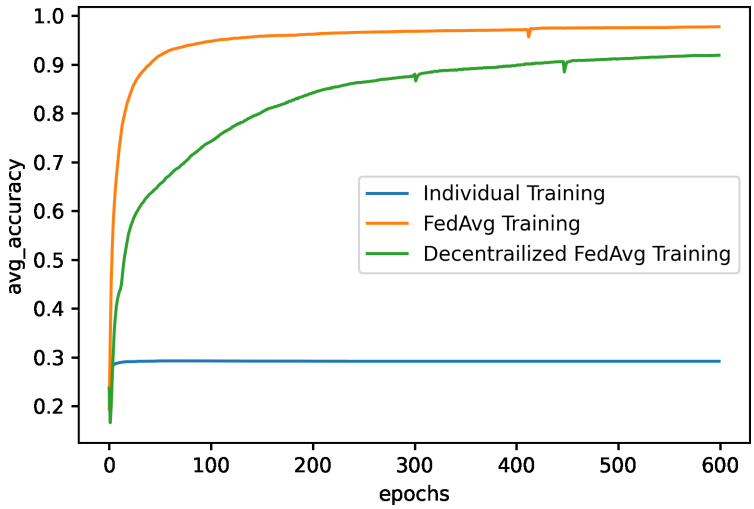

Figure 4.9: Ring Topology Simulation-a

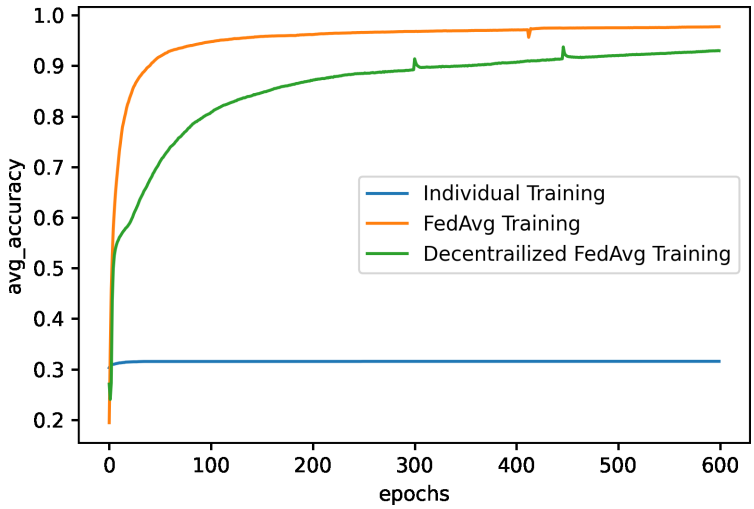

Figure 4.10: Ring Topology Simulation-b

Chapter 5

Conclusion and Future Directions

After the simulations of our experimental results, the below explained are the conclusion to studying all the different topologies associated with Decentralized, Federated Learning. In this chapter, here I am concluding our experimental results comparing the topologies or architectures that improved the performance of our decentralized algorithm. After conducting all the above-demonstrated experiments, the accuracy for each worker when done with one worker excluded, taken three for training is not much efficient; a heterogeneous version is much more unstable with jumps. Later, when the same experiment conducted with the fixing in the jumps by rolling back gave a constant accuracy after certain epochs, which is not desired. Then, moved on to the mesh topology, where this round of iterations achieved better accuracy and less loss. This topology does not sum up to good accuracy, however. Finally, when tried the same process by just changing the topology with the ring architecture. In this ring architecture, each node is connected to the other node once in a ring fashion or a round-robin fashion compared to the computer networks. The accuracy obtained here is better than all the other methodologies mentioned above and architectures. By this, I came to a conclusion that the accuracy for a Decentralized Federated Learning algorithm can be similar to the accuracy of a centralized Federated algorithm with

much more privacy by a central serve free mechanism. The major observation in my contributions is that both the decentralized mesh and ring topologies are outperforming the Federated model. The trade-off between both of them depends on the choice of the application or organization. Mesh topology will be a better choice if the communication cost is not primary. In a similar way, ring topology will be best suited for training models where time is not a big constraint. Both the topologies give us the best results when it comes to the accuracy convergence levels and obtaining less loss of the parameters while training the model. The future work can be extended over this methodology by increasing the number of workers that participate in the training process and the number of iterations in the training process. By scaling this methodology, high accuracy for the model can be achieved, thereby paving an excellent path for moving towards the central management free training and computations of the Machine Learning models.

Bibliography

[Ahamed and Farid, 2018] Ahamed, F. and Farid, F. (2018). Applying internet of things and machine-learning for personalized healthcare: Issues and challenges. In *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*, pages 19–21.

[Bonawitz et al., 2016] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2016). Practical secure aggregation for federated learning on user-held data.

[Cao et al., 2019] Cao, D., Chang, S., Lin, Z., Liu, G., and Sun, D. (2019). Understanding distributed poisoning attack in federated learning. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 233–239.

[Douceur, 2002] Douceur, J. R. (2002). The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer.

[Hu et al., 2019] Hu, C., Jiang, J., and Wang, Z. (2019). Decentralized federated learning: A segmented gossip approach.

[Lian et al., 2017] Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. (2017). Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1705.09056*.

[McMahan et al., 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR.

[Pang et al., 2021] Pang, J., Huang, Y., Xie, Z., Han, Q., and Cai, Z. (2021). Realizing the heterogeneity: A self-organized federated learning framework for iot. *IEEE Internet of Things Journal*, 8(5):3088–3098.

[Truex et al., 2019] Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning.

[Wei et al., 2020] Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q., and Poor, H. V. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469.

[Yang et al., 2019] Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19.

[Zhang et al., 2019] Zhang, J., Chen, J., Wu, D., Chen, B., and Yu, S. (2019). Poisoning attack in federated learning using generative adversarial nets. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 374–380.