Privacy-preserving learning on edge devices (Federated Learning)

Aniket Kumar

Reg No. CS20M002 | cs20m002@iittp.ac.in

22nd March, 2022

Abstract

Federated learning is a new form of machine learning technique which allows a central model to learn from shared local model while keeping all the training data on local machine itself.[1] A local machine can be mobile phone, watches, etc. These days data and privacy is prime concern, by using these techniques we can ensure privacy and we still achieve good accuracy.

Many modern applications in the area of smart computing are based on machine learning techniques. To train machine learning models, a large amount of data is usually required, which is often not readily available at a central location. Federated learning enables the training of machine learning models from distributed data sets at client devices without transmitting the data to a central place, which has benefits including preserving the privacy of user data and reducing communication bandwidth. [2]

This report will explore on different verticals of achieving federated learning, mainly the works compromises of approaches like proxy data driven and weighted averaging. Ways of weight update of central modal to achieve better accuracy and find pattern if any.

Midterm MTP Report

Dept of Computer Science and Engineering
Indian Institute of Technology Tirupati

Evaluation Pannel

		Grading	Signature
Supervisor:	Dr Kalidas Yeturu		
Member:			
Member:			
Member:			
Member:			

Contents

1 Introduction

Traditional machine learning approaches require centralizing the training data on single machine or in a data-centre. This process has a disadvantage in terms of privacy or data of users who provide the training data. All this data is directly visible to one central entity that collected it. Also, all the data is stored at a single location where the model will be trained therefore there is a chance of a single point of failure due to data loss or theft, hazard, natural disaster, etc... also in terms of an organization; this is expensive; maintaining and running the data-centre for 24 X 7. So, to solve this problem, a new form of machine learning technique is evolved called Federated learning or distributed learning. Federated Learning enables edge devices to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the cloud. This goes beyond the use of local models that make predictions on mobile devices (like the Mobile Vision API and On-Device Smart Reply) by bringing model training to the device as well. [1]

The distributed machine learning algorithm is a multi-nodal system that builds training models by independent training on different nodes. Having a distributed training system accelerates training on huge amounts of data. When working with big data, training time exponentially increases, making scalability and online re-training. Compared to distributed learning, federated learning algorithms are fundamentally different and primarily address data privacy. In a traditional data science pipeline, the data is collected to a single server and used to build and train a centralized model. In effect, federated learning has a centralized model using decentralized model training. In federated learning systems, a seed parameter set is sent to independent nodes containing data and the models are trained on the local nodes using data stored in these respective nodes. Once the model is trained independently, each of these updated model parameters are sent back to the central server, where they are combined to create a highly efficient model. Using global data training improves the model efficiency by a significant factor. This also ensures that the data in each node adheres to data privacy policies and protects against any data leak/breach.[2]

2 Motivation

This motivation is data privacy and achieving approximate accuracy as traditional machine learning settings. With increasing data breaches splashed across front-page news, companies have good reason to take security seriously. Federated learning is a powerful technique, yet it amazes without disclosing the data. We tried to do some hands-on and wanted to observe whether it made some sense. Moreover, we observe great results, which drive us to further work on this.

Also in the healthcare domain, for example, histopathology has seen widespread adoption of digitization, offering unique opportunities to increase the objectivity and accuracy of diagnostic interpretations through machine learning. Digital images of tissue specimens exhibit huge heterogeneity from the preparation, fixation, and staining protocols used at the preparation site, among other factors. This variety leads to integration of medical data across multiple institutions is essential. However, the centralization of medical data faces regulatory obstacles, as well as workflow and technical challenges, including managing and distributing the data. The latter is particularly relevant for digital pathology since each histopathology image is generally a gigapixel file, often one or more gigabytes in size. Distributed machine learning in the form of federated setting on decentralized data could be one possible solution to overcome these challenges and promote the adoption of federated learning in healthcare while keeping data secret and similar highly regulated domains.[4]

3 State of the art and gaps

Federated learning is a new form of distributed machine learning framework for collaborative model training with a multi-nodal system of clients (edge devices like mobiles, smartwatches, wearables, etc.). FL offers default client privacy by allowing clients to keep their sensitive data on local devices and only share local training parameter updates with the central server. However, recent studies in weight update have shown that even sharing local parameter updates from a client to the central server may be susceptible to gradient leakage attacks and intrude on the client's privacy regarding its training data. A principled framework is introduced for evaluating and comparing different forms of client privacy leakage attacks. This framework also measures, evaluates, and analyses the effectiveness of client privacy leakage attacks under different gradient compression ratios when using communication efficient FL protocols. [3]

Differential privacy in Federated Learning is new state of the art. Although client data never leaves client devices but Federated Learning is still susceptible to breaches of data privacy [Melis et al., 2019, Bhowmick et al., 2018]. Differential privacy has been combined with Federated Learning to train centralized models with a guarantee of privacy for all clients that participate [McMahan et al., 2018]. By ensuring that gradient updates are not overly reliant on the information in any single training example, gradients can be aggregated centrally with a DP guarantee [Abadi et al., 2016].

4 Problem statements

- To study federated learning behaviour on selective list of popular and important datasets
- To study federated learning behaviour using weight averaging and proxy data driven approach on selective list of popular and important datasets
- To study pattern of central model weight update
- To demonstrate a real world federated learning behaviour on large number of local clients devices (based on natural data cluster)

5 Experimental Setup

Federated Learning is simply the decentralized form of machine learning. In typical machine learning environment data is sent to centralized model but in federated learning the centralized model gets transported to the data. The training happens at the participant's end and only the model characteristics i.e. parameters are forwarded to update the central model. FL enables real time predictions without the internet, security benefits are provided as the data remains in the participant's device also the hardware requirements are less.

Unlike distributed learning where the dataset available to each of the devices is homogeneous, in FL each participant's training data is different from each other's (heterogenous dataset). The purpose of federated learning is to have an ecosystem of different participants whereas the purpose of distributed learning is to improve the speed of computation. The flow of federated learning is from local devices to the central whereas for distributed learning it is from the central to the local devices.

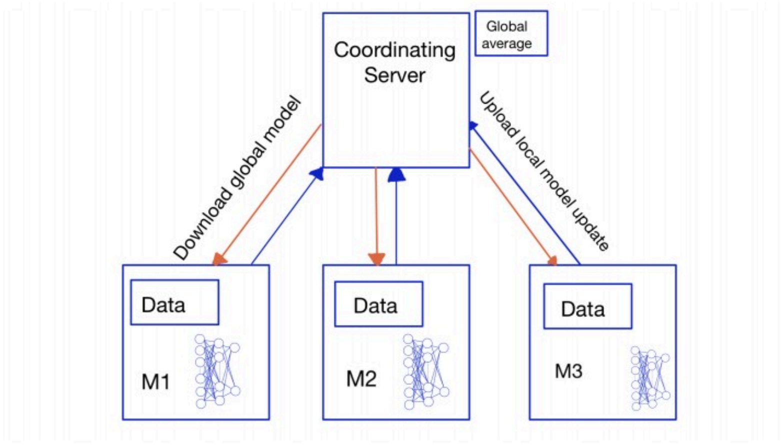

Figure 1: Federated Learning

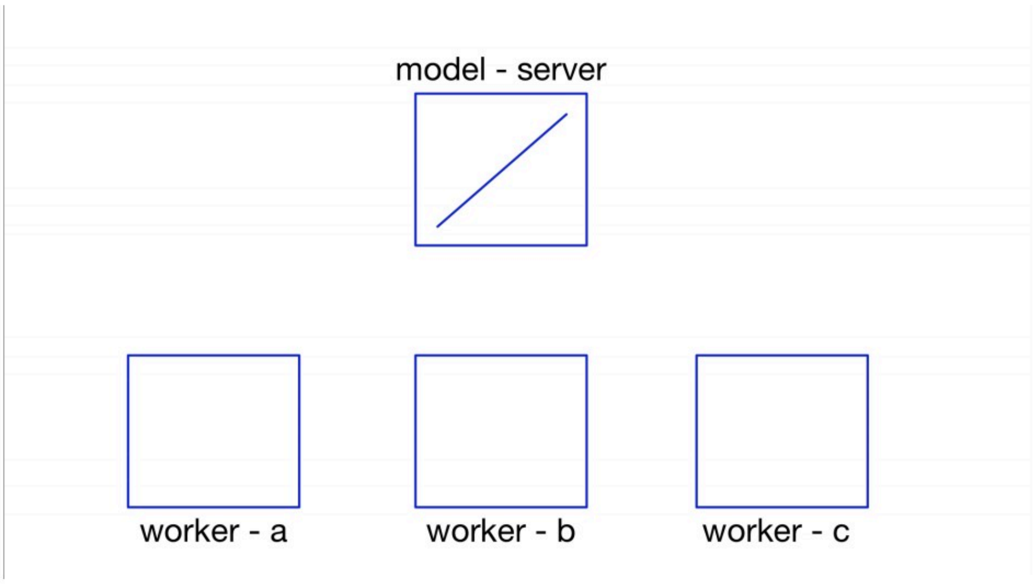

Figure 2: Central model chooses a statistical model to be trained

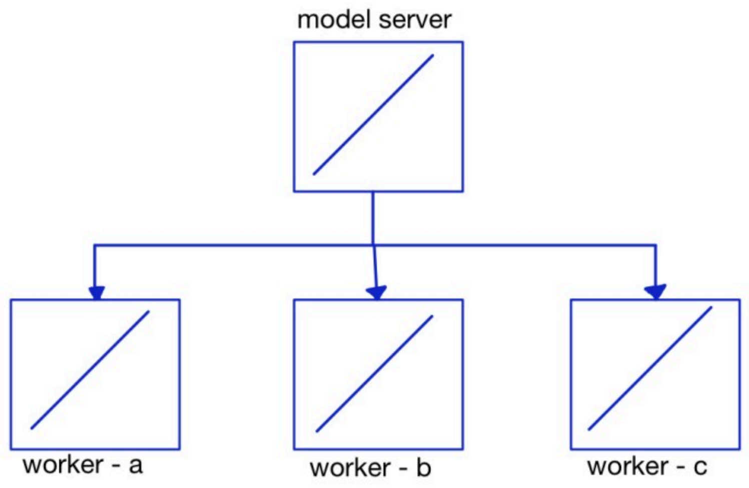

Figure 3: Central server transmits the initial model to several nodes

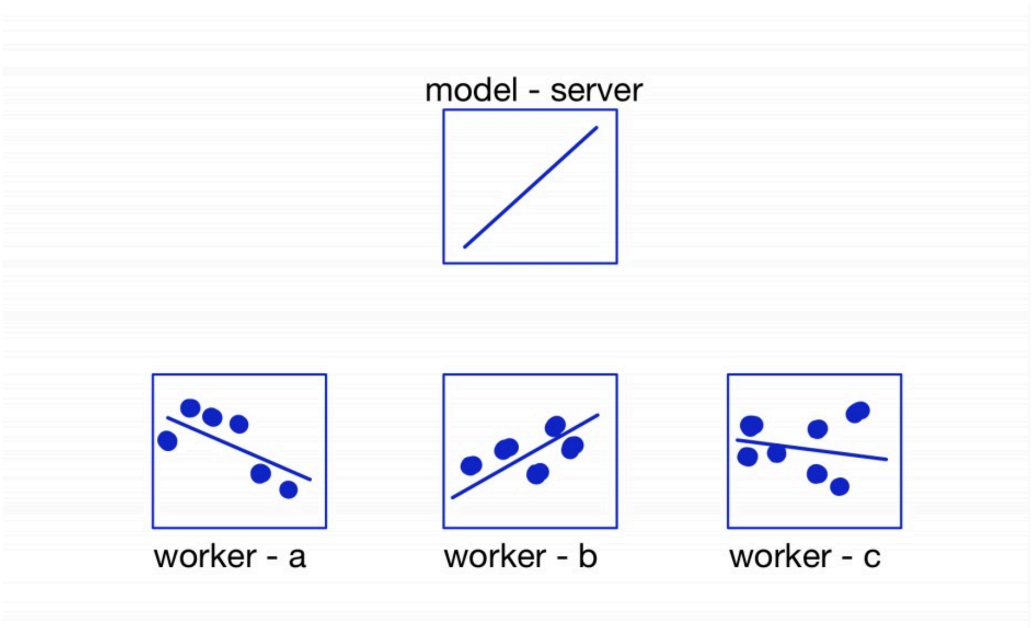

Figure 4: Nodes train this model on it's own local data

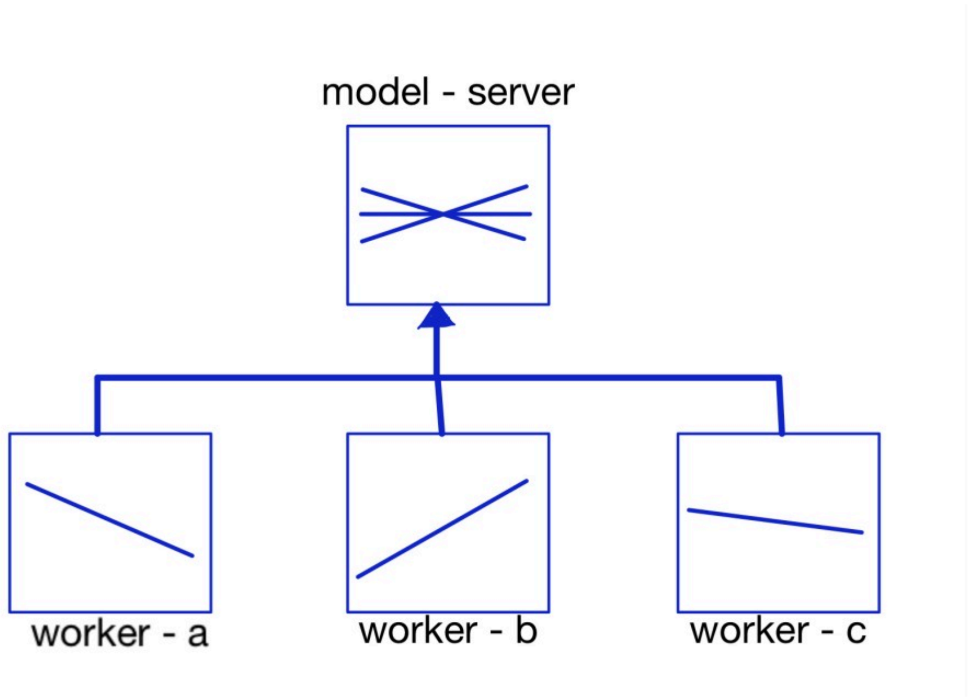

Figure 5: Central server aggregates all the result to generate one global model

6 Proposed methodology

There are two methods by which the central model can be updated.

1. By sending synthetic data to the central model
2. By sending aggregated averaged weights to the central model

The following experiments shows the effect of the above two methods on the performance of the central model. The experiments are performed on the MNIST digit and CIFAR10 datasets.

6.1 Sanity test of GMMs as proxy data in FL setting

6.1.1 Aim

To study the effect on the performance of a central model on adding synthetic data sampled from local GMMs.

6.1.2 Expected outcome

Accuracy of the centralized model formed by adding synthetic data should be more than the model created in absence of the synthetic data. The local models should get enriched.

6.1.3 Observed outcome

Accuracy of the centralized model on local data sets after adding synthetic data is more than its accuracy when the model was not trained on synthetic data points. The updated centralised model have also lead to an improvement of the local models.

6.1.4 Methodology

Algorithm 1 Central model update using synthetic data points

```
Sample D ← MINST digits

Label _index = []
for i € [0,1,..9] do
    for j € [0,....,length(D)] do
        extract label specific index & append to Label _index
    end for
end for
```

```
Central dataset: $C_{data} \leftarrow \emptyset$

for i ∈ [0,1,…9] do
        $C_{data} \leftarrow C_{data}$ U extract (first 5 data points with label i using
Label_index)
end for

$C_{model} \leftarrow$ Train( $C_{data}$ )

for i ∈ [0,1,…9] do
        Create $LocalDataset_i$
        $LocalModel_i \leftarrow$ Train( $LocalDataset_i$ )
        $Accuracy_i \leftarrow$ Score( $C_{model}$ , $LocalDataset_i$ )
end for

// Creating GMMs & Sending to central server
for i ∈ [0,1,…9] do
        $GMM_i \leftarrow$ CreateGMM ( $LocalDataset_i$ )
end for

// Sampling synthetic data
SyntheticDataset $\leftarrow \emptyset$
for i ∈ [0,1,…9] do
        SyntheticDataset $\leftarrow$ SyntheticDataset U $SampleFormGMM_i$
end for

$C_{data} \leftarrow C_{data}$ U SyntheticDataset
$C_{model} \leftarrow$ Train( $C_{data}$ )

for i ∈ [0,1,…9] do
        $Accuracy_i \leftarrow$ Score( $C_{model}$ , $LocalDataset_i$ )
end for
```

6.1.5 Results

Digit Set (Local Clients)	Accuracy Before GMMs	Accuracy After GMMs
0	98	100
1	84	84
2	95	95
3	92	96
4	92	92
5	76	91
6	96	97
7	94	98
8	70	84
9	84	92

Table 1: Accuracy of Central Model on Local Clients before and after updating using GMMs(synthetic data)

6.1.6 Data privacy issue handled due to synthetic data

One of the basic principle of federated learning is data privacy. None of the client is aware of each other's dataset neither is the central model. The central model receives encrypted data which is use to make sure that the central model and no other clients gets to learn each other's data. To maintain privacy rather than sending real data to the central model synthetic data is send which acts like proxy data.

6.1.7 Inferences

Accuracy of the central model increases this implies that our GMMs is working good as proxy data of our local clients. But, while creating the GMMs for our local clients, number of components are fixed which is hyperparameter and essential for the learning of GMMs. So, further we are looking for finding the optimal number of clusters for each of the dataset using silhouette.

6.1.8 Conclusion

We see an increase in the central model accuracy for local clients. But, we can't say for sure it always be, since we are using a toy dataset to demonstrate the federated learning application. So, to further prove its concrete evidence we will try on real world datasets and compare results with federated averaging.

6.2 Sanity test of weight average mechanism

6.2.1 Aim

To observe the impact of weight update of central model based on averaging of NN models of local models and to verify if passing the parameters can lead to a better accuracy for the central model

6.2.2 Expected outcome

Accuracy of the centralized model after weight update will increase, as updates will contain learning of local datasets.

6.2.3 Observed outcome

The performance of the neural network is very low when simple averaging methodology is used. As neural networks are non-linear models there is no guarantee if the simple averaging will improve the performance of the central model or not.

6.2.4 Methodology

Algorithm 2 federated averaging

```
Central dataset: $C_{data} \leftarrow \emptyset$

for i $\in$ [0,1,…9] do
    $C_{data} \leftarrow C_{data}$ U extract (some data points with label i)
end for
```

```
$C_{train}$ , $C_{test}$ ← Split( $C_{data}$ )
$C_{model}$ ← Train( $C_{train}$)
OriginalAccuracy ← Score( $C_{model,}$ $C_{test}$ )

for i € [0,1,…9] do
    $Model_i$ ← $C_{model}$ // Creating 10 copies of $C_{model}$
end for

WeightSum ← Init( shape( $C_{model}$ ), 0 ) // assign shape of $C_{model}$ with all 0's

for i € [0,1,…9] do
    Create $LocalDataset_i$
    Train ( $LocalModel_i$ , $LocalDataset_i$ )
    WeightSum ← WeightSum + Weight( $LocalModel_i$ )
end for

AverageWeights ← WeightSum / Number of $LocalModel_i$
$C_{model}$ ← ReconfigureWeights ( $C_{model}$ , AverageWeights )
AveragedAccuracy ← Score( $C_{model,}$ $C_{test}$ )
Compare( OriginalAccuracy, AveragedAccuracy )
```

6.2.5 Result

1. Accuracy of neural network before averaging is 85%
2. Accuracy of neural network after averaging is 12%

6.2.6 Inferences

Neural network is a non-linear model, simple mean averaging may not always lead to satisfactory results. Simple averaging method will give excellent results if the model used is linear.

6.2.7 Conclusion

There is no clear understanding of why simple averaging fails while weights update on Neural Networks.

6.2.8 Example showing failure of simple averaging of weights in NN

$D = \{(1,1), (-1,1)\}$
$f(x) = \max(0, w \cdot x)$
$L(f(x), y) = (f(x) - y)^2$
$D_1 = \{(1,1)\}$, $D_2 = \{(-1,1)\}$,
On D_1, $w = 1$ On D_2, $w = -1$
Average, $w = 0$
On D_1 , $\max (0, 1 \times 0)$ On D_2, $\max(0, -1 \times 0)$
Error when $w = 0$, 2 units
Error when $w = 1$, 1 units
On the combined data set, $w = 1$ or $w = -1$ has lesser error
On the combined data set, $w = 0$ has higher error
that means, averaged weight is having higher error
This is in case of non-linear classification

6.3 Simple comparison of weight average vs GMM approach

6.3.1 Aim

To compare weight average approach against GMM approach for central model performance

6.3.2 Expected outcome

GMM approach should give better performance

6.3.3 Observed outcome

GMM approach has given 10% better accuracy than weighted averaging of NN

6.3.4 Methodology

Algorithm 3 federated averaging and proxy data for model update

```
Central dataset: $C_{data} \leftarrow \emptyset$

for i $\in$ [0,1,…9] do
```

```
        $C_{data}$ ← $C_{data}$ U extract (some data points with label i)
end for

$C_{train}$ , $C_{test}$ ← Split( $C_{data}$ )
$C_{model}$ ← Train( $C_{train}$)
OriginalAccuracy ← Score( $C_{model,}$ $C_{test}$ )

for i ∈ [0,1,…9] do
        $Model_i$ ← $C_{model}$ // Creating 10 copies of $C_{model}$
end for

WeightSum ← Init( shape( $C_{model}$ ), 0 ) // assign shape of $C_{model}$ with all 0's

for i ∈ [0,1,…9] do
        Create $LocalDataset_i$
        $GMM_i$ ← CreateGMM ( $LocalDataset_i$ )
        Train ( $LocalModel_i$ , $LocalDataset_i$ )
        WeightSum ← WeightSum + Weight( $LocalModel_i$ )
end for



// Sampling synthetic data
SyntheticDataset ← Ø
for i ∈ [0,1,…9] do
        SyntheticDataset ← SyntheticDataset U $SampleFormGMM_i$
end for

$C'_{model}$ ← $C_{model}$

AverageWeights ← WeightSum / Number of $LocalModel_i$
$C_{model}$ ← ReconfigureWeights ( $C_{model}$ , AverageWeights )
AveragedAccuracy ← Score( $C_{model,}$ $C_{test}$ )

$C_{data}$ ← $C_{data}$ U SyntheticDataset
$C'_{model}$ ← Train( $C_{data}$ )

AveragedAccuracy ← Score( $C'_{model,}$ $C_{test}$ )
```

6.3.5 Result

	GMM	NN
Mean Accuracy after update	78.58	70.55

6.2.6 Conclusion

Instead of sending gigabytes of data as weights to central server, which itself is a problem, we can send GMMs and achieve better results.

7 Timeline and Execution Plan

7.1 Final Submission

- Sanity test of local clients as number of clusters in a particular dataset in FL setting
- Compare cycle iteration of weight update of central model using FAA and GMMs
- Final report preparation
- Elsevier Pattern Recognition Letters

8 Conclusion

The experiments result in federated learning settings is giving us satisfactory numbers. We can clearly see that GMMs based approach gives better accuracy over weighted averaging. Moreover, while we are dealing with proxy data we have to communicate less number of parameters to the central server which is significant because in the other scenario we are communicating gigabytes of data. Transfer of huge amount of parameters is not suitable on edge devices since low bandwidth and less computational power problems may arises. Since Neural network is a non-linear model, simple mean averaging may not always lead to satisfactory results.

But the setup is constraint with some ideal scenario like, all of distributed communication scenario is ideal. The datasets we worked on are toy datasets. We worked on CIFAR 10 dataset as well and clearly saw that simple averaging results in same accuracy over cycles, which proves non-linear model are not suitable for this task.

9 Acknowledgement

I would like to express my special thanks of gratitude to Dr. Kalidas Yeturu, Ankit Jauhari and Aditi Palit who always gave me guidance and helped me to know about this new learning.

9 References

[1] G. Research, “Federated learning: Collaborative machine learning with-out centralized training data.”

[2] https://www.kdnuggets.com/2021/11/difference-distributed-learning-federated-learning-algorithms.html

[3] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, “A framework for evaluating gradient leakage attacks in federated
learning,” arXiv preprint arXiv:2004.10397, 2020.

[4] Kalra, Shivam, et al. "ProxyFL: Decentralized Federated Learning through Proxy Model Sharing." *arXiv preprint arXiv:2111.11343* (2021).

[5] McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." *Artificial intelligence and statistics*. PMLR, 2017.

[6] Li, Tian, et al. "Federated optimization in heterogeneous networks." *Proceedings of Machine Learning and Systems* 2 (2020): 429-450.

[7] Sattler, Felix, et al. "Robust and communication-efficient federated learning from non-iid data." *IEEE transactions on neural networks and learning systems* 31.9 (2019): 3400-3413.

[8] H. Zhu, R. Wang, Y. Jin and K. Liang, "PIVODL: Privacy-preserving vertical federated learning over distributed labels," in IEEE Transactions on Artificial Intelligence, doi: 10.1109/TAI.2021.3139055.