Characterizing the Impact of Communication Latency in 5G-enabled FL Systems

Momin Ahmad Khan
Urja Giridharan
Wenkai Dong

ABSTRACT

Federated Learning is an emerging distributed machine learning paradigm where multiple parties, termed *clients*, collaboratively train a joint machine learning model by only sharing their local models and keeping their training data private.

Sharing local models is an essential step in FL, and the communication channel's capabilities constrain it. Since prior works primarily run FL in a simulation setting, FL's performance under real-world network conditions, especially with mobile communication channels, is yet to be explored.

In this paper, we characterize FL under different mobile network conditions and show how various factors, such as network signal strength, latency tolerance, and data heterogeneity, influence the accuracy and time of completion of an FL system.

ACM Reference Format:
Momin Ahmad Khan, Urja Giridharan, and Wenkai Dong. 2024. Characterizing the Impact of Communication Latency in 5G-enabled FL Systems. In . ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Machine Learning (ML) and its applications have become an integral part of our everyday lives. From its most common use in next-word prediction on our smartphone keyboards, to aid in medical diagnosis, and to city-wide facial recognition security systems - machine Learning is being used in almost every aspect of our life. In the past decade, there has been extensive research in the field of machine learning, ranging from improvement in its raw performance, designing attacks to compromise the ML system, countering these attacks in the form of defenses, and offering Machine Learning as a system service by large companies such as Google, IBM, etc.

In recent years, there has been a lot of concern regarding the privacy of user data after encountering several major data breaches, including the 2018 Facebook data leak that exposed the data of 87 million users. Also, the data required to train a neural network usually does not come from a single source, rather it is collected from several sources. In conventional centralized machine learning, all the collected data is pooled onto a single device usually referred to as a *server*. This means that the participants are trusting the central entity with their sensitive data. The server represents a single point of failure in that if an adversary gains access to the server, it will get access to not only all the participant's training data, but the trained machine learning model as well.

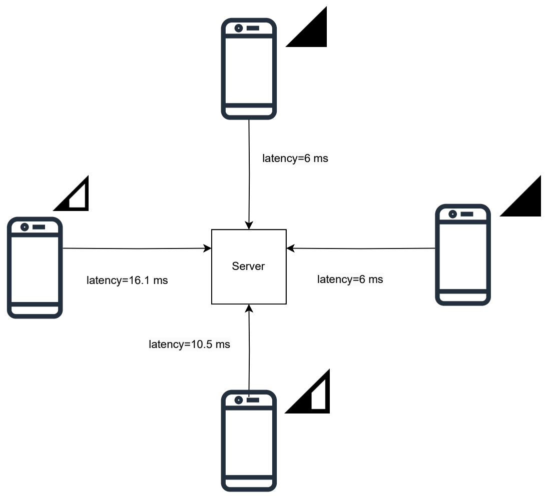
In order to increase the privacy of machine learning systems, Google [8] introduced a novel way of decentralized, privacy-preserving machine learning where the data never leaves the participants. The key idea is to train separate models on each of the participant devices, and only share the model updates with the central entity which is a server that performs aggregation of these shared updates. This technique, coined as **Federated Learning (FL)**, has risen in popularity due to recent laws that aim to preserve and secure data privacy such as GDPR (2018), California Consumer Privacy Act (2020), etc. Prominent distributed platforms such as Google's Gboard [1] for next-word prediction, Apple's Siri for automatic speech recognition [10], and WeBank [15] for credit risk predictions, have adopted this mechanism. Its intrinsic characteristic of promoting collaboration while preserving privacy has rendered it indispensable in various applications, notably in medical diagnosis [4, 6, 11], activity recognition [3, 9, 12, 16], and next-character prediction [13].

Since Federated Learning requires the sharing of models between clients and the server, it is constrained by the network channel's capabilities for this communication. While one can argue that in small FL systems where all the clients are on the same network cluster, the latencies during server-client communication are negligible, the same argument cannot be made for large-scale FL systems where individual clients are edge-devices that require wireless and long-distance connections [17]. In a large-scale FL system with edge devices that send model updates through a 5G mobile network, communication latency plays an important role in the system's performance. This issue is not thoroughly explored in literature, unlike bandwidth constraints, which have been tackled by gradient compression [7], low-rank updates [5], and quantization techniques [14].

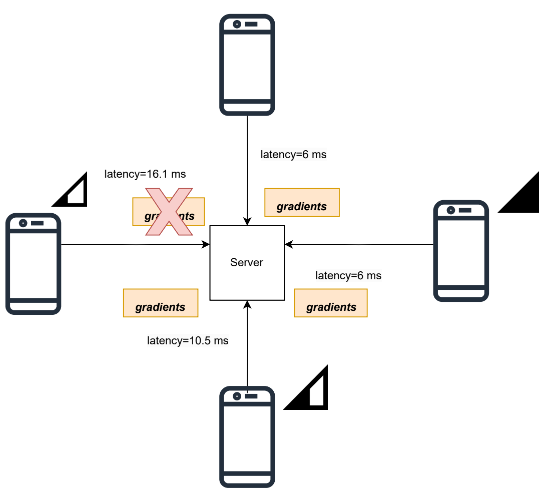
Our Contributions: In this work, we attempt to characterize the performance of an FL system with varying network conditions. We believe that our work is unique as previous works mostly run FL in a simulation setting or consider synthetic latencies. On the other hand, we measure actual latencies associated with running an FL system. Concretely, our main contributions are:

- we first attempt to characterize the impact of varying network conditions and FL settings on the speed and accuracy of an FL system.
- we further analyze the relation of accuracy and time to completion each with network latency tolerance threshold, and data distribution heterogeneity.

Conference'17, July 2017, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn



(a) Model aggregation with no thresholding on client latencies.



(b) Model aggregation with thresholding on high-latency clients.

Figure 1: Edge Devices as FL clients. We use this setting for our analysis. Each client is an edge device, such as a mobile phone and uses its 5G network to upload and download model updates in each round of FL training.

2 BACKGROUND

2.1 Federated Learning

Federated Learning is a decentralized machine learning mechanism where each client has the same copy of a neural network and its own training data. In each round of training, each client computes forward and backward passes over its model to calculate updated weights. It then sends the updated weights to a central server where all other clients pool their updated weights as well. The central server then performs an aggregation algorithm, such as FedAvg [8], and computes a global update. This global update is nothing but the average of all the clients' individual weights. This global update is then shared with each of the clients who start the next round of training with the same weights. This is performed continuously until the model converges. Federated learning thus achieves model training without sharing local training data with other clients or the server involved in the training process.

In a standard deep learning setting, there exists a data-set D, with data points $(d_1, d_2, d_3, ...d_n)$. For each data point, there is a class label that denotes which category that data point belongs to. The class label can be denoted by y_i and it is part of a set Y. The last layer of a neural network will therefore have a number of nodes equal to $|Y|$. These data points are fed into a neural network that is characterized by its parameters which are also called weights and can be denoted by w. The objective is to produce weights w such that the neural network correctly predicts the class of an input data point.

For this purpose, we use a loss function L that computes the error of the output for a given input to the neural network. The loss for a training process is given by $L = \frac{1}{n}\sum_1^n L(w, d_i)$.

The objective of training a neural network is to minimize the loss of the neural network for the given data set. A common technique for performing this optimization is SGD (Stochastic Gradient Descent). It updates the weights of the neural network in each round such that the model finally converges to the lowest loss point.

In federated learning, model convergence is achieved in a different manner from the conventional centralized machine learning. There exists more than one client in the training process. Each client has its own private training data-set D_i. Therefore, there are N local models and a subset k of them are trained in parallel by each client on their private training data-set. The selection of k clients from the pool of N available clients is random. In each round r, they send their weights w_i to a server that performs an averaging function over all the collected weights. The averaged weights for a training round are $w_r = \frac{1}{k}\sum_i^k w_i$. This results in a global model and it is sent back to each client. This global model becomes the new local model for each client and serves as a starting point for the next training iteration. There are R training rounds and the final set of weights when model convergence is achieved is w_R. This is the result of performing federated averaging over local updates at the server to achieve a single model having global accuracy G_{acc}.

3 METHODOLOGY

In this section, we outline the methodology of our experiments for characterization. Figure 1 shows the high-level idea of our system. The system comprises various client devices, each executing various machine-learning models locally. These devices could range from smartphones and tablets to IoT devices and edge computing nodes. The system operates within a 5G network, where each client device experiences differing levels of signal strength. This variation in signal strength affects the latency, i.e., the time it takes for data to travel between the client device and the central server over the 5G network. Thus, the latency for each of these devices would be different. The central server is a hub for aggregating model updates (gradients) received from the client devices. These updates represent the incremental changes made to the local models during their training process. The server collects these updates and performs aggregation techniques. Once the gradients are aggregated, they are sent again for further rounds of training. To ensure maximum accuracy, the server waits for all client devices to send their model updates before proceeding with the aggregation process. This approach guarantees that the latest information from all devices is considered during aggregation, leading to a more comprehensive and accurate model update. However, this waiting time can be prolonged by clients with higher latency, delaying the overall training process. Conversely, to expedite the training process and minimize completion time, the system may opt to prioritize clients with lower latency, dropping or giving less weight to updates from clients with higher latency. This approach accelerates the aggregation process by reducing waiting times, but it may sacrifice the accuracy of the aggregated model since updates from certain devices are omitted or given less priority.

Data collection: We wrote a Python script to measure the latency it took for a client to send data to the server. The latency is observed during data transmission, and measured at regular intervals during the data transfer process. The latencies for different signal strengths were collected by measuring the latencies in different locations. For poor signal strength, we went to the basement. Most areas on campus had moderate signal strength.

Our experimental setup was designed to replicate realistic network conditions, incorporating variations in latency and signal strength. The script would start by configuring the experimental setup, including setting up the client and server instances of the iperf tool. During each round of data collection, the script would initiate data transmission between the client and server using the iperf tool thereby recording the latency. The latency values observed during each interval for each signal strength condition were logged and stored in a CSV file.

Each interval value represents the latency (in seconds) observed during a specific time interval of the data transmission. This time was recorded for each client for each round. Initially, a fixed latency was used for each client for all rounds. In the second phase of the project, different latencies were recorded for each client for each round to simulate the real-world scenario.

These latencies were recorded for various network conditions, i.e., we recorded the latency with high signal strength, medium signal strength, and poor signal strength. In this project, our objective was to assess the influence of latency on Federated Learning (FL) performance within a 5G network context, aiming to comprehend how fluctuating network conditions impact FL training in real-world scenarios.

Datasets: We ran our experiments by utilizing two primary datasets: MNIST and FashionMNIST. The MNIST dataset comprises 60,000 single-channel handwritten images of digits ranging from 0 to 9, each with dimensions of 28x28 pixels. Additionally, it includes a testing set of 10,000 images with identical specifications. The FASHIONMNIST dataset comprises 60,000 training examples and 10,000 test examples of grayscale images, each measuring 28x28 pixels, categorized into 10 classes.

We found that MNIST is more resilient under high latency conditions. This is because MNIST is a much simpler dataset and even if some clients drop out of the training process, it does not hurt the accuracy much. We further analyzed the impact of various latency thresholds on MNIST and FashionMNIST.

FL settings: To distribute the data among 20 clients, we employed the FANG distribution method. Each client participated in 100 rounds of training. To control the data's heterogeneity and observe its impact on accuracy, we adjusted the bias parameter governing the degree of heterogeneity, setting it to values of 0.1, 0.5, and 0.9.

Hyperparameters used for training on both the datasets: The learning rate was set to 0.01, Adam was used as the optimizer, and a batch size of 32 was utilized during training.

The experiments were conducted by varying the following parameters:

1. Varying the latency tolerance: We varied the threshold for dropping clients during each epoch to examine its effect on FL performance. By adjusting this threshold, we aimed to balance the trade-off between accuracy and the duration required for the server to collect information from clients. Lower thresholds result in higher accuracy but longer wait times for data collection, while higher thresholds expedite the process but may sacrifice accuracy.

2. Varying the network condition: We selected different threshold values for different RSRP signals.This threshold varied for data recorded for different RSRP signals.

3. Varying Data Heterogenity: We ran our experiments selecting 5 bias values : 0.1, 0.3, 0.5, 0.7, and 0.9. We fix the other parameters and observed the change while varying the bias. High bias values imply higher heterogenity of the data.

4 CHARACTERIZATION

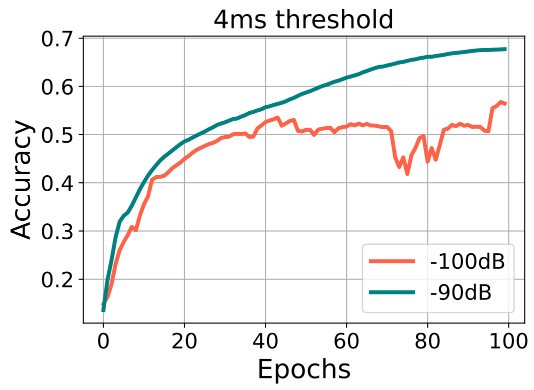
In this section, we conduct several experiments to analyze the performance of the FedAvg algorithm when latency becomes an important consideration during aggregation. Specifically, this section will scrutinize the impact of dropping high-latency clients. Since we are dropping clients based on their latencies, we need to define a latency threshold. We will also study the impact of different latency thresholds for different network conditions. Finally, we study the impact of a very important hyperparameter in FL: the amount of heterogeneity.

4.1 Varying Network Conditions

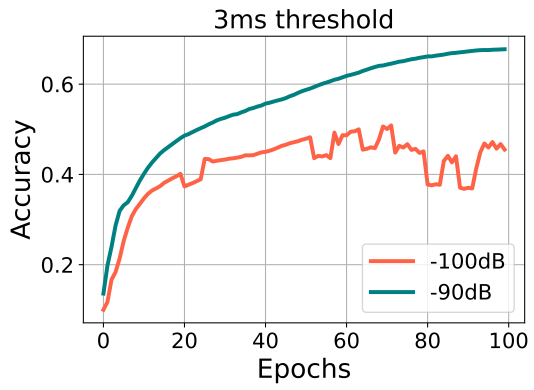
Here, we record the latencies of different clients using a 5G hotspot and see their performance in an FL setting. One set of readings was taken with the RSRP at -100dB (2 signal bars) and the other at -90dB (3 signal bars).

Justification for our choice of network conditions: We tried recording latencies at 1 signal bar, but the iperf connection broke in that area. Also, finding a place on or near campus with 4/4 signal bars where the RSRP was fairly stable was hard. So, we chose 2 and 3 signal bar areas for our analysis because we could get stable RSRPs there.

Comparision between -90dB and -100dB: We run our FedAvg algorithm with 10 clients and drop those clients whose latency exceeds a certain threshold in each round. In Figure 2, we show the test accuracy at every epoch for two network conditions: RSRP of -90dB and the other having -100dB. We show our results for latency thresholds of 3ms and 4ms. We can see that the clients in a



(a) 4ms



(b) 3ms

Figure 2: Performance of FedAvg at -90dB and -100dB for latency thresholds of 3ms and 4ms respectively.

-90dB setting suffer greatly and are unable to reach decent accuracy. Specifically, at -90dB, with a threshold of 4ms, none of the clients is dropped as they all have latency less than 4ms. This results in an accuracy of 56% for the -100dB setting, while the -90dB setting reaches a much higher accuracy of 67%.

4.2 Varying Latency Tolerance

Our latency tolerance ensures that clients with latency higher than our latency threshold are not selected in the aggregation process for that round. Thus, it becomes an important hyperparameter in our experiments. Here, we study the impact of varying this latency threshold to see how much it impacts the system's overall accuracy.
Accuracy is sensitive to latency threshold: In Figure 3, we see the impact of different latency thresholds on the global model test accuracy. We chose three values, 2ms, 3ms, 4ms, for this experiment, and we can see that the global model struggles to achieve decent accuracy when the threshold is low. This is because when the threshold is low, for example, 2ms, all clients with a latency greater than 2ms will be dropped in that round. We compare each threshold accuracy with the case when we do not have any threshold, and as expected, the no threshold accuracy is greater in all cases. However, this comes with a tradeoff. Although our graphs show that a high

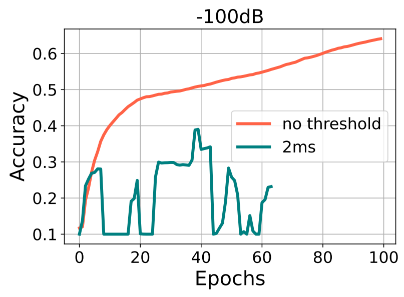
Table 1: Total delay encountered when applying different thresholds for bad(-100dbm) and good(-90dbm) network conditions.

rsrp(dbm)	threshold(ms)	total delay(s)
-90	1	0.4
-90	1.2	0.58
-90	1.4	0.86
-100	2	0.47
-100	3	1.26
-100	4	1.75

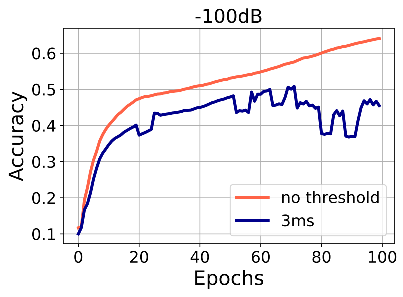
threshold results in a higher accuracy, we can see from Table 1 that a high threshold leads to a much higher delay in the time to completion. In real-world settings, such a delay could be significant enough to impact the overall performance of the system.
Threshold selection depends on network conditions: In Figure 4, similar to the previous experiment, we vary the latency threshold for a network condition having RSRP -90dB. We can see from the figure that the global model accuracy also reduces as we reduce the threshold. However, the key observation here is that with better network conditions, we can select a much stricter latency threshold. For example, in the -90dB scenario, we can achieve an accuracy of 49% with a threshold of 1ms, but in the -100dB scenario, we are only able to achieve an accuracy of 39% with a much higher threshold of 2ms. We also tried running an experiment where we kept the threshold 1ms in the -100dB scenario, but the model failed to train at all as most of the clients were dropped.

4.3 Varying Data Heterogeneity

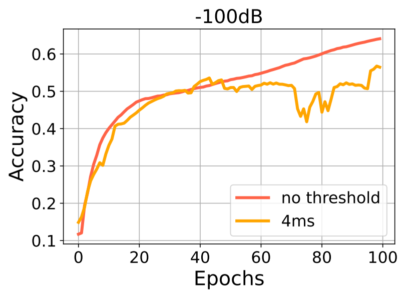
Data heterogeneity is one of the most important factors influencing the performance of an FL system. Our analysis would have been incomplete without observing the impact of different heterogeneous data distributions. Our hypothesis is that dropping clients will have a higher impact under high heterogeneity because now the dropped clients might have had more samples of a certain class. Therefore the resulting global model will have missed out on crucial information at the cost of achieving aggregation in less time.
High heterogeneity has a higher impact on client dropping: We select our network condition as -100dB and fix the latency threshold to 4ms. We only vary the heterogeneity bias value and run our experiment for five bias values: 0.1, 0.3, 0.5, 0.7, 0.9. A higher bias value corresponds to a a higher heterogeneity level and vide versa. From Figure 5, we can see that at higher heterogeneity levels, we lose a lot on accuracy when we use our client dropping approach. Note that in all five of these experiments, the same clients are dropped every round. But the difference is that the different amount of information is being lost in each of these. The most amount of information is being lost in the bias=0.9 scenario because here data distribution is highly skewed and dropping one client has a higher chance of losing out on most samples of a particular class. Based on this result, we emphasize that network conditions must not be overlooked in FL, especially when operating in real-world scenarios.



(a) 2ms

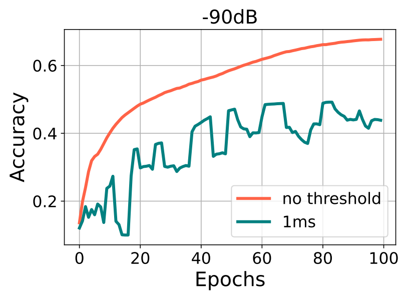


(b) 3ms

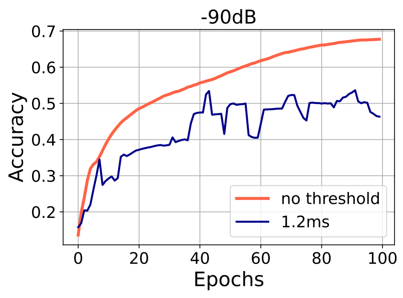


(c) 4ms

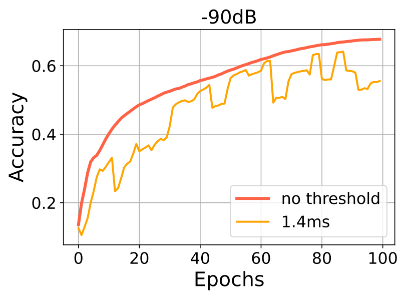
Figure 3: Clients in a -100dB network setting: Comparison of the impact of different latency tolerance thresholds on the performance of our global model.



(a) 1ms



(b) 1.2ms



(c) 1.4ms

Figure 4: Clients in a -90dB network setting: Comparison of the impact of different latency tolerance thresholds on the performance of our global model.

4.4 Key Takeaways

In summary, we have three key takeaways from our set of experiments:

(1) Good network conditions result in high global model accuracy and less time to complete FL training. This is because less clients are dropped so we lose less on accuracy. Also, each client has low latency, thereby leading to low completion time of training.
(2) High latency tolerance leads to high accuracy but more time to complete FL training. This is because we are allowing more clients into the aggregation phase because we have a high latency threshold value, but now it takes more time to complete FL training as we have to wait for more clients to be aggregated.
(3) High heterogeneity leads to low accuracy but does not impact time to complete FL training. This is because when clients are dropped in high heterogeneity settings, the dropout of a client impacts training more as it contains skewed class samples, i.e., it probably has more samples of a certain class than most clients. This leads to a phenomenon called *local model drift*, where local models drift away from each other and the aggregated global model is not the best representation of all of the clients' local data. On the other hand, in a homogeneous setting, the global model is a near-perfect representation of the clients' local data as all of the clients possess a similar data distribution.

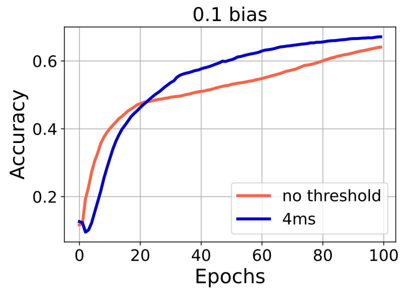
5 LIMITATIONS AND FUTURE WORK

Our work is a humble first attempt to characterize the impact of mobile network conditions on the performance of an FL system. While we have been able to demonstrate the role of network signal strength, latency threshold, and data heterogeneity in influencing the accuracy and time of completion of an FL system, our analysis has certain drawbacks and limitations. Here, we discuss some of those.

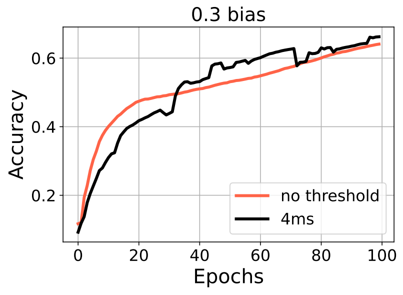
Although our study is motivated by large-scale real-world FL systems, we do not perform our analysis over such a multi-million-device system and leave it for future work. Ideally, we should be conducting our experiments with actual geo-distributed devices so that we get an accurate assessment of mobile networks around the world.

We are using standard benchmark datasets like MNIST and FashionMNIST for this study. We acknowledge that these are *toy* datasets and we only used them for proof of concept. Ideally, in our future work we should be using some large-scale naturally-distributed real-world datasets such as those in [2].

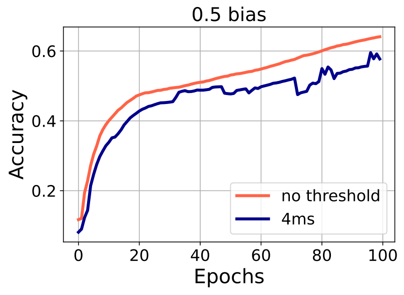
Additionally, our study is limited to only one network provider: Mint Mobile. For our future work, we should consider different



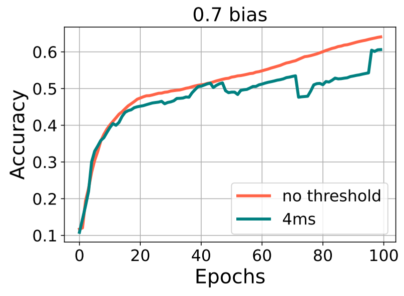
(a) $\alpha = 0.1$



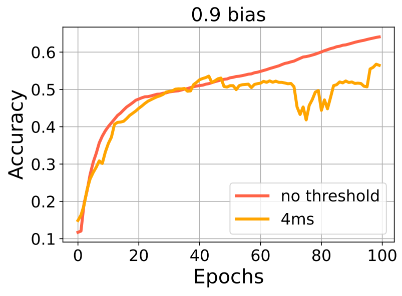
(b) $\alpha = 0.3$



(c) $\alpha = 0.5$



(d) $\alpha = 0.7$



(e) $\alpha = 0.9$

Figure 5: Clients in a -100dB setting. Comparison of accuracy between no-threshold and 4ms threshold for different heterogeneity values. Heterogeneity is represented by α. A higher α means a higher heterogeneity.

network service providers, both inside and outside the US. Another thing to consider would be the location of the server. For now we only used one iperf server. But it would be interesting to see how the location of the iperf server affects system performance. One way we can reduce overall latencies and improve global model accuracy is to do a profiling of the userbase and optimally select the server location so that it is closer to places with more users.

Another limitation of our work is that we only performed analysis over the FedAvg [8] algorithm. There are many more FL algorithms in literature, and we will be using those for our future works. It would be interesting to compare different FL algorithms and observe their resilience to varying network conditions.

6 CONCLUSION

In this study, we characterized the impact of network conditions on the performance of mobile FL systems. Specifically, we showed that low signal strength leads to high client latencies and we either have the choice to drop them out of the aggregation phase or wait for them and incur a long training completion time. Furthermore, we show the role of an important hyperparameter, the latency tolerance threshold, in impacting the accuracy and training time completion of FL. Lastly, we demonstrate that while heterogeneity does not impact time of completion, it significantly impacts the accuracy when we are adopting the client-dropping policy. We present three key takeaways that will help us and others in future research in this field.

REFERENCES

[1] Federated learning: Collaborative machine learning without centralized training data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.
[2] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.
[3] Sannara Ek, François Portet, Philippe Lalanda, and German Vega. Evaluation of federated learning aggregation algorithms: application to human activity recognition. In *Adjunct proceedings of the 2020 ACM international joint conference on pervasive and ubiquitous computing and proceedings of the 2020 ACM international symposium on wearable computers*, pages 638–643, 2020.
[4] Ines Feki, Sourour Ammar, Yousri Kessentini, and Khan Muhammad. Federated learning for covid-19 screening from chest x-ray images. *Applied Soft Computing*, 106:107330, 2021.
[5] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *NIPS Workshop on Private Multi-Party ML*, 2016.
[6] Hanchao Ku, Willy Susilo, Yudi Zhang, Wenfen Liu, and Mingwu Zhang. Privacy-preserving federated learning in medical diagnosis with homomorphic re-encryption. *Computer Standards & Interfaces*, 80:103583, 2022.
[7] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv:1712.01887*, 2017.
[8] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.
[9] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. Clusterfl: a similarity-aware federated learning system for human activity recognition. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 54–66, 2021.
[10] Matthias Paulik, Matt Seigel, Henry Mason, et al. Federated Evaluation and Tuning for On-Device Personalization: System Design & Applications. *arXiv:2102.08503*, 2021.
[11] Adnan Qayyum, Kashif Ahmad, Muhammad Ahtazaz Ahsan, Ala Al-Fuqaha, and Junaid Qadir. Collaborative federated learning for healthcare: Multi-modal covid-19 diagnosis at the edge. *IEEE Open Journal of the Computer Society*, 3:172–184, 2022.

[12] Konstantin Sozinov, Vladimir Vlassov, and Sarunas Girdzijauskas. Human activity recognition using federated learning. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 1103–1111. IEEE, 2018.

[13] Jingwei Sun, Ang Li, Lin Duan, Samiul Alam, Xuliang Deng, Xin Guo, Haiming Wang, Maria Gorlatova, Mi Zhang, Hai Li, et al. Fedsea: A semi-asynchronous federated learning framework for extremely heterogeneous devices. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, pages 106–119, 2022.

[14] Peng Sun, Wansen Feng, Ruobing Han, Shengen Yan, and Yonggang Wen. Optimizing network performance for distributed dnn training on gpu clusters: Imagenet/alexnet training in 1.5 minutes. *arXiv preprint arXiv:1902.06855*, 2019.

[15] Utilization of FATE in Risk Management of Credit in Small and Micro Enterprises. https://www.fedai.org/cases/utilization-of-fate-in-risk-management-of-credit-in-small-and-micro-enterprises/, 2019.

[16] Yuchen Zhao, Payam Barnaghi, and Hamed Haddadi. Multimodal federated learning on iot data. In *2022 IEEE/ACM Seventh International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 43–54. IEEE, 2022.

[17] Ligeng Zhu, Hongzhou Lin, Yao Lu, Yujun Lin, and Song Han. Delayed gradient averaging: Tolerate the communication latency for federated learning. *Advances in Neural Information Processing Systems*, 34:29995–30007, 2021.