Analysis of Federated Learning as a Distributed Solution for Learning on Edge Devices

Saba F. Lameh*, Wade Noble†, Yasaman Amannejad†, Arash Afshar‡
*University of Tabriz, Tabriz, Iran. Email: farshbaf.saba98@ms.tabrizu.ac.ir
† Mount Royal University, Calgary, Canada. Email: {wnobl374, yamannejad}@mtroyal.ca
‡Aranite Inc., Calgary, Canada. Email: aafshar@aranite.com

***Abstract*—Sensors and smart devices are continuously collecting massive amounts of data. Today's state-of-the-art machine learning (ML) techniques, such as deep networks, are typically trained using cloud platforms, leveraging the elastic scalability of the cloud. For such processing, data from various sources need to be transferred to a cloud server. While this works well for some application domains, it is not suitable for all due to privacy and network overhead concerns. Sharing life logging photos and videos from cell phones and wearable devices can cause privacy concerns for users. To respond to the needs of such applications, federated learning (FL) is proposed as a distributed ML solution for learning on edge devices, such as cellphones. In FL, data does not leave users' devices. All users collaboratively train a model without sharing their data. Each user trains a local model with their data and shares the model updates with a FL server to aggregate and build a global model. Such training respects users' privacy while reducing the network overhead for transferring data to a central server. In this paper, we aim to study the FL approach for learning on edge devices. We investigate the performance of the FL model with various parameter settings and compare its accuracy and training time to a traditional learning technique that is trained on a central server. For this purpose, we train a convolutional deep learning network for image recognition in traditional and federated format. The results confirm that it is possible to conduct a complex learning task on distributed devices without sharing their data with a central server, and such federated models can achieve comparable accuracy to central models.**

***Keywords*— Distributed Machine Learning, Federated Learning, Performance Analysis, Edge Computing**

I. Introduction

Learning from distributed data over edge devices has gained increasing interest in the recent years. This increase in popularity is because of the following reasons. *First*, there is a need for processing massive amount of data that is continuously being generated through mobile phones, wearable devices, autonomous vehicles. Transferring all the data to a remote cloud server is not always feasible due to limiting factors such as network bandwidth of these devices and the long propagation delays that can incur unacceptable latency. *Second*, the private nature of these data, e.g., life-logging videos and recorded phone calls, causes privacy concerns for sharing the data with cloud servers. *Finally*, the transfer of such massive data to a cloud server for processing can burden the backbone networks especially for applications with unstructured data, e.g., image and video analytics. These factors along with the improvements in the storage and computation capacity of these edge devices are shifting the data processing and model training of ML applications from cloud servers to edge devices. This has led to a growing interest in federated learning (FL) as an intersection of ML and edge computing fields.

Consider large number of user devices, such as mobile phones, each with personal collections of photos or autonomous vehicles with images captured by cameras. If all the distributed data on these devices are accessible in a central place, one can obtain a high-performance ML model that has been trained on an extremely large dataset. However, it is not desirable for clients to share their data due to privacy concerns. FL [1] is introduced recently as a decentralized ML approach suitable for edge computing that responds to such processing needs. In FL, data remains private on personal devices. These mobile devices work cooperatively to train a complex ML model without sharing their data. Each device trains a local model using its own private data and shares their model parameter updates, e.g., model weights, with a FL server which aggregates them into a global model. The FL server sends back the aggregated model to mobile devices for further training. This process continues iteratively in multiple rounds until a desirable accuracy is achieved or a training budget is hit.

FL has recently gained popularity and is shown to be effective in different application areas [2–7]. This growing popularity is due to the advantages that FL provides over conventional ML techniques. FL uses the distributed power of the edge devices to perform its main computations, and since the computation is done close to the user, the data does not need to leave the device and hence provides more user privacy. Moreover, the large unstructured data such as image and video files do not need to leave the user device. Therefore, the network overhead is reduced, and the network latency for applications that require quick decision making may improve.

To use FL in scale, Google recently announced a new TensorFlow Federated (TFF) Learning library. To build FL models with TFF, we need to understand how the parameters of the FL framework affects the performance of a federated ML models. To this end, in this paper, we aim to train a deep learning network for a common image recognition task using FL framework and test the accuracy of the model. We build our model based on MNIST dataset [8] to answer the following research questions (RQs):

978-1-7281-8376-3/20/$31.00 ©2020 IEEE

- *RQ1*: How does the training time grow as a function of the number of clients and the training time?
- *RQ2*: How does the client selection affect the convergence time and the accuracy of the trained model?
- *RQ3*: How is the accuracy and the training time of a FL model compared to a central model?

The remainder of the paper is organized as follows. In Section II, we discuss FL and its properties. Section III discusses related work. Section IV and V describe our experiment setup and results, respectively. We answer RQ1 in Section V-B and V-C, RQ2, in Section V-D and V-E, and RQ3 in Section V-F. Conclusions and future work are offered in Section VI.

II. Federated Learning

The goal of FL is to provide a framework for distributed ML on edge devices. In general, there are two main entities in FL process: the data owners, i.e., $client$, and the global model owner, i.e., $server$. Let $Clients = \{client_1, client_2, ..., client_n\}$ denote the set of n clients, each of which has a dataset D_i, $i \in n$, stored on their personal devices, e.g., cellphones. In classical approaches, all clients send their data to the server and the server trains a conventional model, $model$, in central fashion using all data $D = \cup_{i=1}^{n} D_i$. However, as described earlier, this is not possible for all applications due to privacy concerns and the network bandwidth limitations of the edge computing devices. In FL, clients do not share their private data with the remote server, instead, they build a local model, $model_i$, using their own data, D_i, and share the parameters of their trained model with the FL server. The server collects all local model parameters and aggregates them to build a global model, $model_{federated}$. The trained model is sent back to clients for further enhancements. This process continues for r rounds of training.

A typical training process is shown in Fig. 1 and includes the following four steps:

1) **Model broadcast**: The server defines the structure of the model to be trained by all clients and decides the hyper-parameters of the local and global models, e.g., the optimizer or the learning rate of a neural network. Next, the server broadcasts the model and parameters to clients. In the first round, the broadcasted model is not trained. In the following rounds, the server broadcasts the model aggregated from the local training of clients.
2) **Local model training**: Clients participating in the training process download the model and train the model based on their local data. The objective of each client during its local training is to find optimal model parameters that minimize the model loss based on client's local data.
3) **Model parameter updates**: When each client finishes its training, they share their trained model with the FL server.
4) **Global model aggregation**: When the participating clients share their models with the server, the server aggregates them into a global model. The server's objective is to minimize the global loss function.

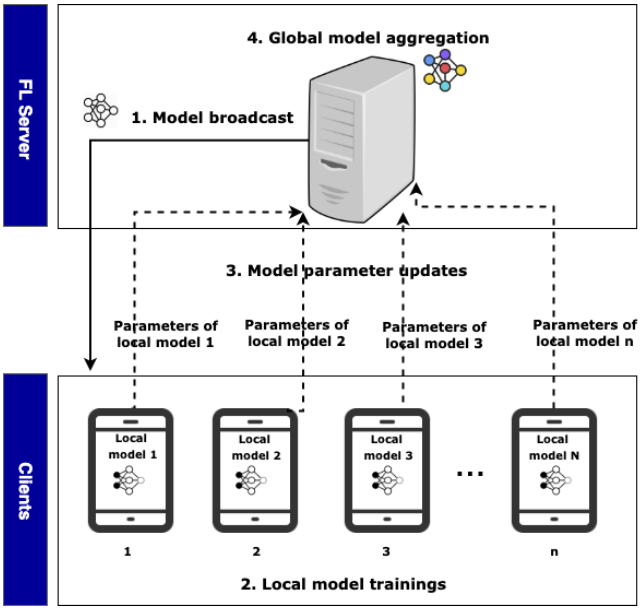

Fig. 1: The main steps in FL training process

Training of a FL model is an iterative process and the training happens in multiple rounds. To respond RQ1, we analyse the accuracy of the model trained for various rounds in Section V-B. In an ideal case, the accuracy of the model trained in this distributed and iterative format, should be close to the accuracy of the model trained in conventional format over all available data in a central place. However, FL model may result in lower accuracy as a trade-off for gaining higher privacy. In Section V-F, we compare a model trained in federated and in central fashion to answer RQ3.

It should be mentioned that during the training process, the number of participating clients may vary. Some clients might not be available to participate in the training process, e.g., their device might be turned off. Therefore, in each round of the training process, the server only expects to get model updates from a subset of all clients, $C \subset Clients$. The participating clients can affect the performance of the global model and therefore client selection is an important topic in FL [9]. To answer RQ1, we analyse the effect of the number of participating clients in Section V-C. We also analyse the effect of the participating clients in terms of their data sizes, and if the same or different clients participate in each round of the training in Sections V-D and V-E to answer RQ2.

As described in FL process, clients do not share their data with the server. Clients only share their local model parameters; this significantly reduces the network overhead compared to when the data moves over the network. In a simple FL protocol, clients and the server exchange the model parameters without any modification such as data compression [10]. In each round, the parameters are exchanged twice, once when clients send their local models

to the server, and once when they download the aggregated model from the server. This is done for all participating clients. The communication cost is a function of model size, the number of rounds and the number of participating clients. This cost can be compared with the cost of exchanging all client data records with a central server. The cost of exchanging data, specially for unstructured data such as image or video records can be very large. This motivates the use of FL as a distributed ML technique that results in higher privacy for clients and lower communication overhead.

III. Related work

Federated Learning is proposed to enable building ML models on edge devices while preserving users' privacy [1]. This learning method is shown to be useful in various application areas such as vehicular networks [3, 4], mobile packet classification [2], and cyberattack detection [5–7]. Programming libraries, simulators, and benchmark datasets are developed to facilitate using this learning framework [11–13]. However, to use FL in large scale, there are security [14], communication [10, 15–17], and performance [12, 18, 19] challenges that need to be addressed.

Many researchers have focused on optimizing the communication overhead in the FL process [10, 15–17, 20]. Konecny et al. [10] proposed two techniques for reducing the communication costs. Their approaches are based on learning from a restricted space parametrized using a smaller number of variables and learning from compressed model. Others have modified the FL protocol to reduce the communication overheads. Wang et al. [15] proposed adding a feedback loop to FL process. In each iteration, clients receive feedback information regarding the global tendency of model updating. Each client checks if its update aligns with this global tendency and is relevant enough to model improvement. By avoiding uploading irrelevant updates to the server, their technique can substantially reduce the communication overhead while still guaranteeing the learning convergence. In [20], before starting the learning process, clients evaluate the relevance of their data and filter out the noisy data that are not relevant to the given FL task.

The need for performing ML tasks on resource constrained edge devices and communicating the model parameters with the server makes performance an important topic in this framework. Researchers have studied the performance of federated operations [12, 18, 19]. Nilsson et al. [18] studied the performance of three federated optimization algorithms, namely Federate Averaging [1], Federated Stochastic Variance Reduced Gradient [10], and CO-OP [21]. Wang et al. have analyzed the convergence rate of distributed gradient descent from a theoretical point of view and proposed an algorithm that determines the best trade-off between local update and global parameter aggregations to minimize the loss function under a given resource budget [19]. Mugunthan et al. [12] has proposed a simulator that supports simulating the accuracy of FL models and the convergence time of the model. Moreover, the simulator is able to model the latency and the real-time drop out of clients. In [22], a profiling tool is proposed that predicts the computation time and the energy consumption of each learning task on mobile devices. This estimation is used to prevent unnecessary computations that will be available after the deadline of the server for aggregating the local models.

Clients participating in the training process of a FL model can affect the final output of the model greatly. Some clients may not be available to participate in some training rounds or the server may decide to select a subset of clients to reduce the communication costs. Many researchers have focused on client selection problem for FL [9, 16, 23, 24]. Nishio and Yonetani [9] proposed a client selection strategy that filters out slow clients based on the estimation of the clients' execution time at the selection stage. The model selects participants based on their computation capabilities to complete FL training before deadline. Since some clients can be slower than others, asynchronous [22, 25] and semi-asynchronous [16] FL protocols are offered to reduce the effect of straggling clients and stale models [26]. Moreover, to motivate clients to participate in the learning process, Zhan et al. [27] have proposed incentive-based protocols for FL process.

While the proposed solutions can be effective in improving the performance of the federated training, the use of these techniques and tuning the parameters of FL depends on understanding the effect of these parameters. To help researchers in this area, we have automated the process of such comparisons [28]. In this work, we investigate the effect of the training parameters, such as the number of training rounds, on training time and the accuracy of the federated model. Moreover, we analyse the effect of clients in terms of the number of participating clients, the impact of their data size, and the changes posed by the same or varying groups of clients' participation in different training rounds. Such analysis of the various effects of the participating clients is not done in previous work.

IV. Experiment setup

A. Experiment Testbed

To do our analysis, we train a federated deep learning network for image recognition task. To implement our FL model and compare it with a central image recognition approach, we use the Google Colab platform [29]. This equips us with 1 GPU and 12 GB of RAM. To implement the deep learning network used in the central and FL models, we use the Python language and the Keras library [30]. For implementing the FL model, we use the TFF library [11]. We have made all the code used in this work publicly available [31].

B. Experiment Factors

Table I lists the various experiment factors and their different levels. The default levels used in the experiments are shown in bold format.

As described earlier, FL is an iterative process. The number of rounds that the training takes can affect the accuracy and the training time of the federated model. To

TABLE I: Experiment factors

Factor	**Value**
Training rounds	r = {5,10,15, 20, 25, 50, **100**}
# of clients	$\|C\|$ = {2, **5**, 10, 15}
Client data size	Smallest, **Default**, Largest
Participating clients	**Constant**, Variable
Learning method	**Federated**, Central

see the effect of the training rounds, we vary the number of rounds from 5 to 100 in 7 steps and record the accuracy and the training time metrics.

To study the effect of clients' participation in training, we change the number of clients in 4 levels from 2 to 15. The default strategy for selecting these clients is to select them randomly from all existing clients. To investigate the effect of selecting clients with small or large data samples, we compare our random client selection strategy against the case where the selected clients have the smallest number of unique data records among all clients and also selected clients that have the largest number of unique records. Fig. 2 shows the distribution of the number of unique data records that each client has. On average, each client has 101 unique data records that are shuffled and used during training. The number of data records that each client owns varies between 9 and 132 records. Clients selected with the smallest and the largest data record strategies have on average 14.6 and 121.6 records, respectively.

In addition to the number of participating clients and the wealth of their data samples, we also investigate the accuracy of a FL model where clients participating in each round varies and we compare it with a model trained by the same group of clients during all training rounds. We refer to the clients selected by these strategies as *variable* and *constant* clients.

We train a deep learning network in federated format to recognize hand-written digits. We use the MNIST dataset [8] which is a writer-annotated handwritten digit classification dataset collected from 3,383 users. In total, the MNIST dataset contains 70,000 grayscale images of size 28×28 for 10 hand-written digits. We use the federated version of the MNIST dataset that is created using the LEAF project [13]. The data we use is available in the simulation package of TFF [32], and is keyed by the original writer of the digits. Since each writer has a unique style, this dataset exhibits the kind of non independent and identically distributed (non-i.i.d.) behavior which is expected from federated datasets. In the federated model, we repeat the data of each client 1,000 times, and shuffle them. In each round of training, data used in each epoch is sampled from this repeated and shuffled dataset.

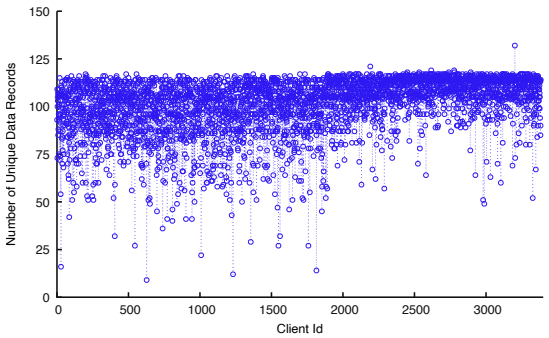

Fig. 2: Number of unique training records of clients

In this work we train a central and a federated model to complete the same image recognition task. For the central network, we use a deep network with 5 layers of convolution, pooling, and dense layers. The architecture of the network is shown in Fig. 3. This network is shown to have a high accuracy on the MNIST dataset [33]. To make the comparison of our federated and central model easier, we use the same network structure and hyper-parameters for both models, e.g., batch size = 50, and epoch = 25. The FL server defines this structure and assigns it to all clients to train it.

C. Experiment Process and Metrics

To evaluate the effect of each parameter on the federated model, we train the model using the data from participating clients. We report the *training time* in seconds and the *training accuracy* as a percentage of the correctly predicted labels by the model on the same data that it is trained on. We also report test accuracy, which is the accuracy of the model when predicting the labels of data records that the model has not seen during the training process. We report the test accuracy both based on the test data of clients participated in the training process, and as a prediction for the test data of all existing 3,383 clients. We repeat each experiment 3 times and report the average metrics.

V. Results

A. Federated Deep Learning for Image Recognition

For training a federated model, we use the same network architecture described in the previous sections. All participating clients train the same model independently and communicate their model parameters with the federated server. The federated server is using stochastic gradient descent optimizer to aggregate the local models.

Fig. 4 shows the accuracy of the model trained with 5 sampled clients over 100 rounds. Each of these 5 clients are training the network for 25 epochs in each round. As it can be seen from the figure, the accuracy of the trained model improves as clients and the server communicate over the rounds of training. From round 27, the accuracy of the training data stabilized around 99%. When using the model on the test data records of the same 5 clients, i.e., handwritten digits by the same clients, the model achieves 90% accuracy. Note that test data records used for each client are new data samples that the model has not seen during the training. The high accuracy on test data of the sampled clients shows that the model is not overfitted to the training data set.

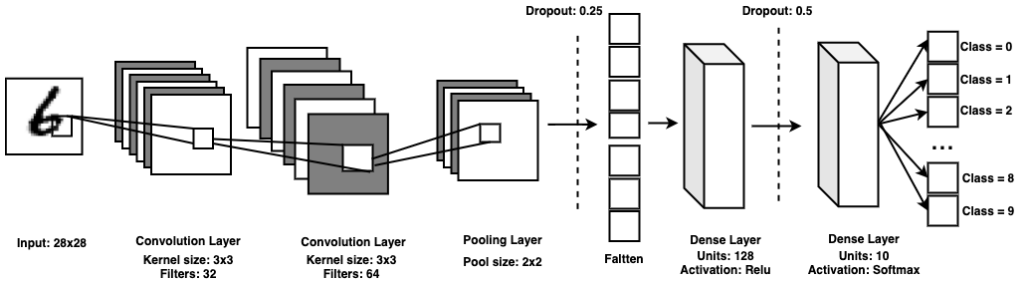

Fig. 3: Network Architecture

We also tested the generalizability of the model trained by these 5 clients to recognize the hand-written digits of all existing clients, i.e., all 3,383. We tested the model on the test data records of these clients. In this case, the model achieved 83% accuracy. This is a high accuracy for a model that has not seen any handwritten samples from most of these clients. In this image recognition task, some of the features that a model can learn are independent from the writer of the digit, and the model still works well on other clients' data. However, since the model has not seen or captured the writer dependant features of unseen clients' hand writings, the accuracy is lower than the previous case. This experiment confirms that our trained model has captured enough features to generalize well and detect the handwritten digits by other clients as well.

B. *Effect of the Training Rounds*

One of the important parameters that can affect the training time and accuracy of a FL model is the number of training rounds, r. In this section, we analyze the effect of the number of rounds on the training time, and also on the model accuracy.

Fig. 5, shows how the training time changes with the number of training rounds. Each round consists of training on a client device, communicating the model parameters with the server, and aggregating the model parameters on the server. The training on the client devices and communicating the model to the server happens in parallel and the time of each round depends on the time that the client with the less computation power and the slowest network bandwidth. Increasing the number of clients increases the chance of having a slower client device, while also increasing the time of aggregation on the server side. In this scenario, all client devices are assumed to be homogeneous, i.e., having the same processing power and the network bandwidth. Therefore, the training time increases linearly with the number of clients. In heterogeneous environments, the time of each client will be dependant on the time of the slowest client in terms of the highest total processing and communication time.

Fig. 4 shown in the previous section presents the accuracy of the model on the training data of the sampled clients from $r = 1$ to $r = 100$. To get the accuracy of the model with certain number of r rounds on the test data, we add checkpoints in the training process to save the model state after certain rounds. We show the test accuracy when using these models on test data of the 5 sampled clients and on all existing 3383 clients.

Fig. 6 shows the results for models trained with 5, 10, 15, 25, 50, and 100 rounds. The model accuracy does not increase linearly with the number of training rounds. The improvement on the accuracy is higher at the beginning and slows down after about 25 rounds in both cases. As can be seen in the figure, when the model is tested on the sampled clients, i.e., 5 clients, a smaller amount of training is enough to get a good model accuracy, e.g., 15 rounds in this case. However, to train a more general model that works for all clients, more rounds of training are required.

When the majority of the client populations participate in training, we can reduce the number of training rounds.

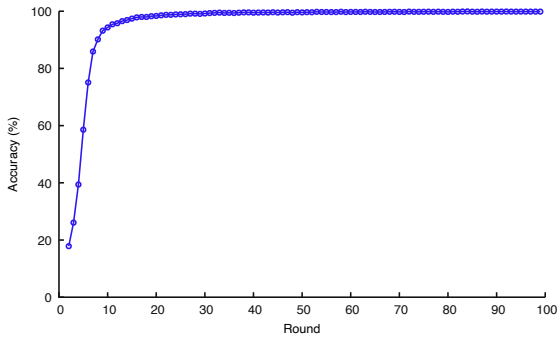

Fig. 4: Training accuracy of a federated model

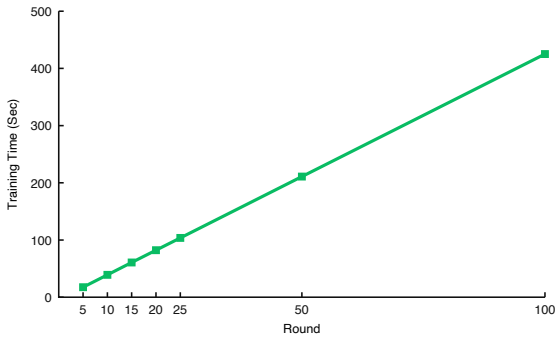

Fig. 5: Training time increase with number of rounds

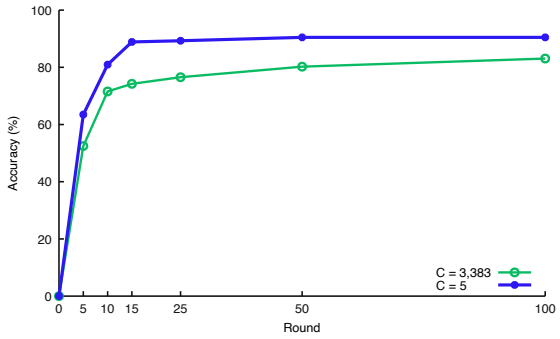

Fig. 6: Test accuracy achieved with different training rounds

However, when a small subset of clients is building a model, they need to spend more time to collaboratively train a model that is generalizable. Training with 5 out of 3,383 clients represents an extreme case where most clients either do not have incentive to participate in the training process, or they are left out because of other reasons such as network instability.

C. Effect of Number of Sampled Clients

As we discussed in the previous section, an important parameter when building a federated model is the number of sampled clients, C. In this section, we select a set of clients from all existing clients and let the train a FL model. We vary the number of sampled clients from 2 to 15 in 4 levels and show the model accuracy on the training data of these clients in Fig. 7, and on the test data of all clients in Fig. 8.

As can be seen in Fig. 7, in all four experiments, the model converges around round 100 and the accuracy gets close to 99% for all cases. The model trained with smaller data samples trains faster and quickly attains training accuracy close to 99%. For example, the model with 2 clients achieves 99% accuracy on training data from around round 20. This is while it takes longer number of rounds for the model with 15 clients to achieve the same accuracy.

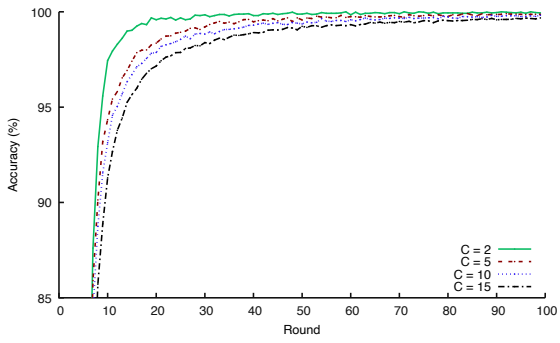

Fig. 7: Training accuracy with different number of clients

While the model with less data quickly achieves higher accuracy, this does not guarantee that the model will work well on unseen data records. To investigate the accuracy of the trained models on new data records, we evaluate all models with the test data of all clients. Fig. 8 shows the results. As it can be seen from the figure, the model trained by two clients does not have a high accuracy when testing on unseen data records. This means that the model has not learned enough features to generalize its image recognition on data points of other clients. When more clients participate in the training, the accuracy of the model improves. With 10 clients, the model accuracy gets to 90% and stabilizes at this point. Since clients can work concurrently, participation of more clients should not affect the time of each round significantly. However, as discussed in Fig. 7, with more clients the model requires more rounds to converge during the training process.

D. Effect of Sampled Clients' Data Size

Not only the number of clients, but also the number of data points that each client owns is important for the accuracy of a FL model. In the previous steps, we randomly sampled the clients. To study the effect of the data points, we use two different strategies in this section for sampling clients. We select 5 clients with the *smallest* total number of data records and 5 clients with the *largest* number of data records. We compare the model trained by these clients with the model trained with our *default* strategy of randomly sampling 5 clients. Fig. 9 shows the number of unique data records that each group has. The 5 clients in the smallest group have in total 73 unique data records, while this is about 480 and 620 for the default and the largest group of selected clients.

We trained our FL model with these client groups. Fig. 10 shows the accuracy of the model during the training rounds. As it can be seen, the accuracy of all models converges at the end of 100 rounds of training. The model trained with clients with the smallest data samples take longer time to achieve a high accuracy. On average, each client in our data set has 101 unique data records. These records are shuffled and used in the training rounds. However, the number of unique data records in the smallest group varies between 9 to 22. These records are not enough to learn all 10 classes of digits in the dataset. The data records in the randomly selected clients varies between 93 to 109, and this allows the client models to view a larger variety of data samples for each class. Therefore, the accuracy of the model trained with this client group is close to the accuracy of the model

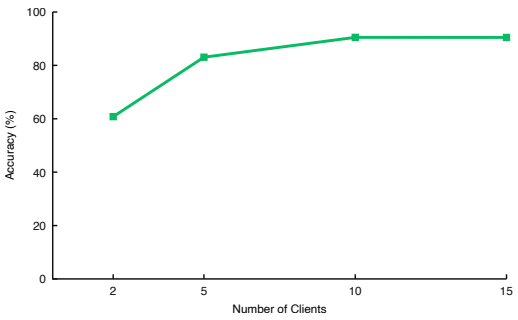

Fig. 8: Test accuracy with different number of clients

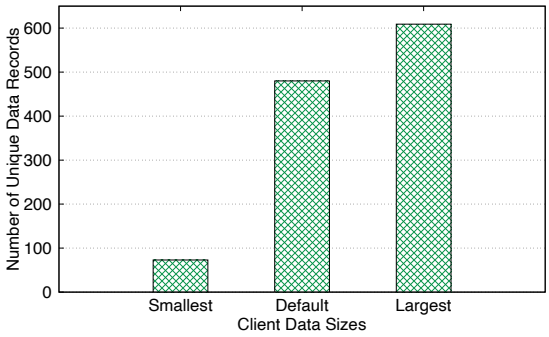

Fig. 9: Unique data records of each client groups

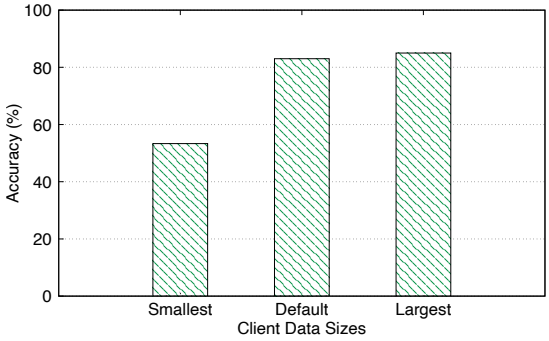

Fig. 11: Test accuracy with different client groups

trained by the clients with the largest data sample where each client has between 118 to 132 records.

To investigate if the trained models with these client groups are generalizable, we evaluated them on the test data of all existing clients. The results are shown in Fig. 11. The model trained with the smallest number of data records has an accuracy around 50%. The accuracy of the model trained by randomly selected clients is close to the accuracy of the group of clients with the largest dataset. This shows that to have an effective client selection strategy, we need to consider both the number of clients and the wealth of data samples that each client has encountered.

E. Effect of Client Participation Strategy

In the previous sections, the sampled clients for performing the training rounds remain the same during the whole training process. To investigate a different scenario, we study the case where clients participating in training may change in each training round. This represent a real case where clients drop out from the training process due to unavailability of their device, e.g., when a cellphone device is turned off, or when the server decides to aggregate local models without waiting for a straggler client.

Fig. 12 shows the training accuracy of the models trained with the same set of sampled clients versus when different client samples participate in training round. As it can be seen from this figure, in both cases, the trained FL models are able to achieve high accuracy. The accuracy of the model trained with different sampled clients is slightly lower than when the clients stay the same in all training rounds.

We also evaluated both models on test data records of all clients. The interesting point we observed is that while the model trained with varying clients has slightly lower accuracy on the training dataset, it achieved a better accuracy when tested on all clients' test data. Fig. 13 shows the results. The model trained with varying client samples achieves 95% accuracy on the test data. This is while the accuracy of the model trained with the same client samples is 83%. This can be due to the fact that when clients vary it allows the model to be trained with a wider variation of training records. Although the model has slightly slower convergence and lower accuracy during training, the output model is more generalizable. This observation along with observations from the previous two sections confirm that client selection is an important factor in FL process and proper client selection can result in better models.

F. Comparing Federated and Central Models

To compare our FL model with a central approach, we use the same network structure trained with each FL client and train it centrally. For this, we assume that all data records are available on a central server. We divide the data into test and train samples in such a way that the total number of training records used to train the central model is almost the same as the total number of data records used by sampled clients in FL model. Hyper-parameters of the network such

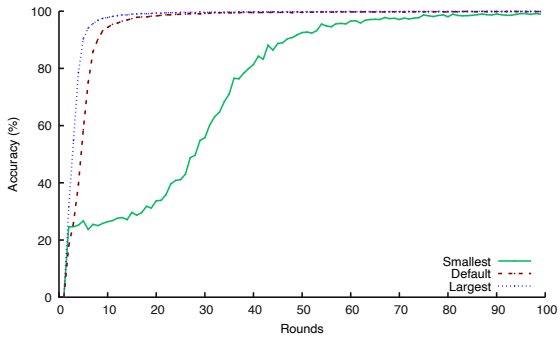

Fig. 10: Training accuracy with different client groups

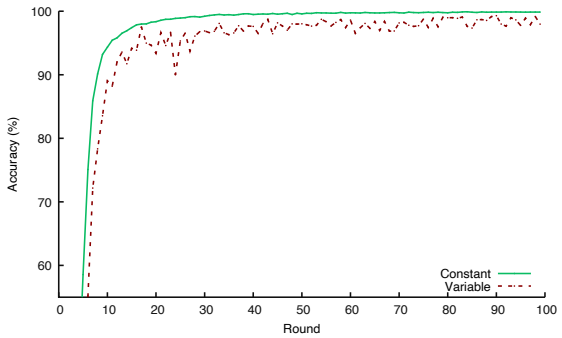

Fig. 12: Training accuracy with constant and varying clients

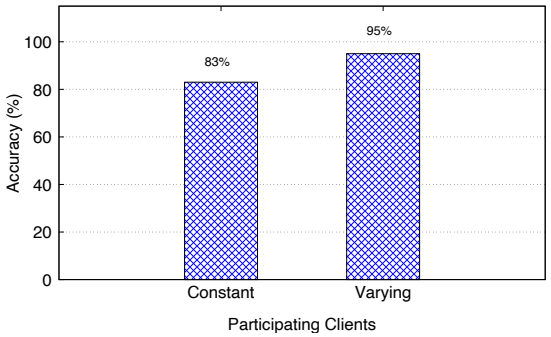

Fig. 13: Test accuracy with constant and varying clients

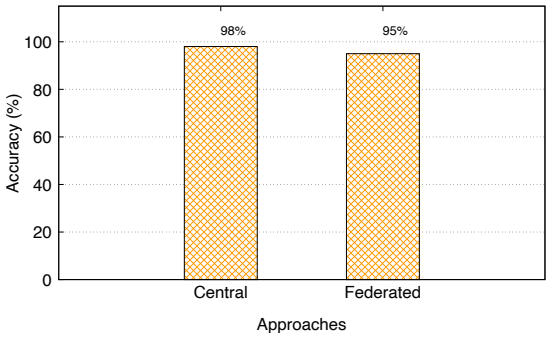

Fig. 15: Test accuracy of federated and central models

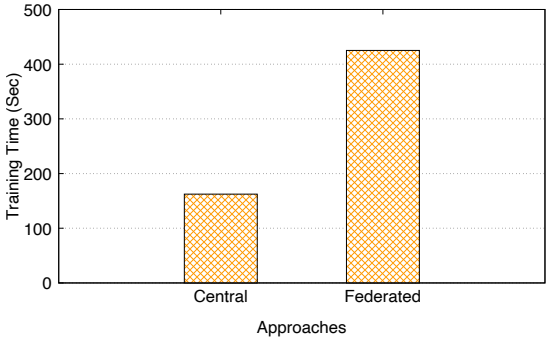

Fig. 16: Training time of federated and central models

as the batch size and the number of epochs are kept the same in both models.

We trained the central model for 25 epochs and have recorded the accuracy of the model while training in Fig. 14. As can be seen in the figure, the accuracy of the model increases and reaches 98% on the training records. We then, evaluate the trained model on our test dataset. The test data set is about 30% of the original dataset that the model has not seen in the training process. The model achieves 98% accuracy on the test data set as well, confirming that the model is well trained and is not over-fitted to the training dataset.

Fig. 15 and Fig. 16 show the test accuracy and the training time of both models, respectively. As it can be seen in Fig. 16, with the same model structure, the FL model achieves slightly less accuracy than the central model. This is because in the FL model data is distributed over the client devices, and we do not have access to all data to train a global model. Instead, the global model is built by aggregating the local models which results in slightly lower accuracy. As discussed before, the loss in the accuracy of FL model can be reduced by increasing the number of participating clients or increasing the training rounds. From Fig. 16 we can see that training a FL model takes a longer period of a time. This is a trade-off between the data privacy, i.e., not sharing client data records with a server and the communication cost savings that FL can have. Moreover, using distributed techniques such as FL is inevitable when data is sufficiently large that the storage and computation cost of sending data to a central server is impractical.

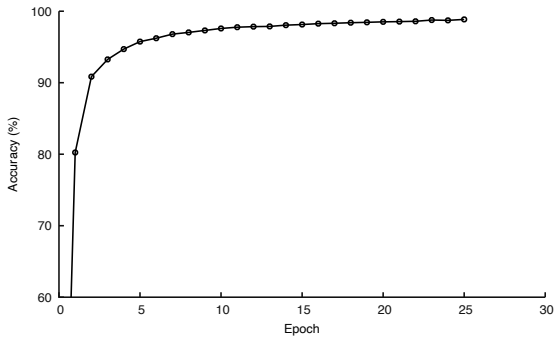

Fig. 14: Training accuracy of the central model

VI. Conclusions and Future Work

FL allows distributed users to collaboratively train a model without sharing their private data. This enhances user's privacy and reduces the communication overhead. In this paper, we trained a deep learning model for an image recognition task in federated format and evaluated how the training time and the model accuracy changes with federated parameters. We show the importance of proper selection of these parameters. Specifically, we show with three experiments the effect of participating clients and the importance of client selection strategies in FL models. Moreover, we compared our model with a deep learning model with the same network structure trained centrally. We show that FL, while improving the privacy and communication overhead, is able to achieve a comparable accuracy to a central model. In the future, we plan to expand our analysis by training more models and considering device heterogeneity for client devices.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

[2] E. Bakopoulou, B. Tillman, and A. Markopoulou, "A federated learning approach for mobile packet classification," *arXiv preprint arXiv:1907.13113*, 2019.

[3] Y. Liu, J. James, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet of Things Journal*, 2020.

[4] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency v2v communications," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2018.

[5] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, M. D. Mueck, and S. Srikanteswara, "Energy demand prediction with federated learning for electric vehicle networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[6] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 756–767, IEEE, 2019.

[7] A. Abeshu and N. Chilamkurti, "Deep learning: the frontier for distributed attack detection in fog-to-things computing," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 169–175, 2018.

[8] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," 2010.

[9] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2019.

[10] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[11] "Tensorflow federated learning." https://github.com/tensorflow/federated. Accessed: 2020-06-30.

[12] V. Mugunthan, A. Peraire-Bueno, and L. Kagal, "Privacyfl: A simulator for privacy-preserving and secure federated learning," *arXiv preprint arXiv:2002.08423*, 2020.

[13] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[14] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948, 2020.

[15] W. Luping, W. Wei, and L. Bo, "Cmfl: Mitigating communication overhead for federated learning," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 954–964, IEEE, 2019.

[16] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. A. Jarvis, "Safa: a semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Transactions on Computers*, 2020.

[17] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *International Conference on Artificial Intelligence and Statistics*, pp. 2021–2031, 2020.

[18] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, pp. 1–8, 2018.

[19] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 63–71, IEEE, 2018.

[20] T. Tuor, S. Wang, B. J. Ko, C. Liu, and K. K. Leung, "Overcoming noisy and irrelevant data in federated learning,"

[21] Y. Wang, "Co-op: Cooperative machine learning from mobile devices," 2017.

[22] G. Damaskinos, R. Guerraoui, A.-M. Kermarrec, V. Nitu, R. Patra, and F. Taiani, "Fleet: Online federated learning via staleness awareness and performance prediction," *arXiv preprint arXiv:2006.07273*, 2020.

[23] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-fl: Cooperative learning mechanism using non-iid data in wireless networks," *arXiv preprint arXiv:1905.07210*, 2019.

[24] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," *arXiv preprint arXiv:1905.10497*, 2019.

[25] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 21–28, Springer, 2018.

[26] H. H. Yang, A. Arafa, T. Q. Quek, and H. V. Poor, "Age-based scheduling policy for federated learning in mobile edge networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8743–8747, IEEE, 2020.

[27] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet of Things Journal*, 2020.

[28] Y. Amannejad, "Building and evaluating federated models for edge computing," in *Proceedings of the International Conference on Network and Service Management (CNSM 2020)*, IEEE, in-press.

[29] "Google colab." https://colab.research.google.com/. Accessed: 2020-07-10.

[30] "Keras." https://keras.io/. Accessed: 2020-07-10.

[31] "Federated Learning Experiments." https://github.com/Yasaman-A/federated-learning-experiments.git. Accessed: 2020-09-10.

[32] "Tff simulation package." https://www.tensorflow.org/federated/api_docs/python/tff/simulation. Accessed: 2020-07-10.

[33] "Simple mnist convnet." https://keras.io/examples/vision/mnist_convnet/. Accessed: 2020-07-10.