2022 IEEE 6th International Conference on Fog and Edge Computing (ICFEC) | 978-1-6654-9524-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICFEC54809.2022.00015

iSample: Intelligent Client Sampling in Federated Learning

HamidReza Imani, Jeff Anderson, and Tarek El-Ghazawi
Department of Electrical and Computer Engineering, The George Washington University, Washington, DC 20052 USA
hamidreza,jeffa,tarek@gwu.edu

***Abstract*—The pervasiveness of AI in society has made machine learning (ML) an invaluable tool for mobile and internet-of-things (IoT) devices. While the aggregate amount of data yielded by those devices is sufficient for training an accurate model, the data available to any one device is limited. Therefore, augmenting the learning at any of the devices with the experience from observations associated with the rest of the devices will be necessary. This, however, can dramatically increase the bandwidth requirements. Prior work has led to the development of Federated Learning (FL), where instead of exchanging data, client devices can only share weights to learn from one another. However, heterogeneity in device resource availability and network conditions still impose limitations on training performance. In order to improve performance while maintaining good levels of accuracy, we introduce iSample. iSample, an intelligent sampling technique, selects clients by jointly considering known network performance and model quality parameters, allowing the minimization of training time. We compare iSample with other federated learning approaches and show that iSample improves the performance of the global model, especially in the earlier stages of training, while decreasing the training time for both CNN and VGG by 27% and 39%, respectively.**

***Index Terms*—federated learning, heterogeneous systems, resource constrained devices, edge computing, machine learning**

I. Introduction

The ever increasing use of artificial intelligence (AI) has made machine learning (ML) applications an important part of our lives. ML applications typically require a large amount of data to train models with sufficient accuracy. Despite the large amounts of available data from edge devices and Internet of Things (IoT) sensors as a whole, individual devices may not have enough data at their disposal to successfully train a model.

However, aggregating data from multiple devices can be problematic and raise several challenges. The size of training data for natural language processing or computer vision coupled with the large number of client-to-client transfers makes client-based data sharing protocols unattractive from a network performance perspective. Moreover, this approach will decrease the scalability of the system due to the fact that various types of devices participate in training with different availability and operating times. In addition, sharing the training data can raise privacy issues when the applications are dealing with personal user information such as healthcare data or emails.

To address the network and scalability issues, centralized learning approaches such as [1] were proposed in which all local data is transferred to a central server for model training. While this approach can decrease the network load, it cannot minimize the load to levels acceptable by commonly-used wireless protocols. Additionally, these approaches ignore privacy issues surrounding data sharing among individual devices.

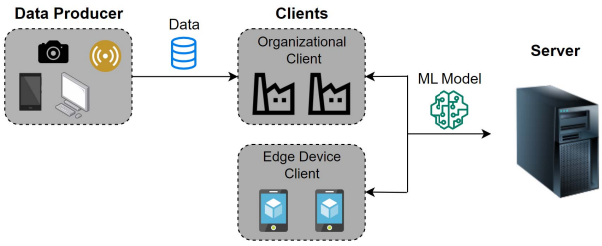

Fig. 1: FL aggregates models trained on disparate data sets to derive a unified model used for client update.

To address the aforementioned issues, Federated Learning (FL), which is a decentralized communication-efficient ML paradigm, was proposed to preserve data privacy of the users participating in the training. In FL based on [2], an initial model is sent to the clients, and then in every iteration, a subset of clients train their ML model using their private data, then send their models to a central server, central server aggregates the models and sends the averaged model back to the clients. Finally, the clients replace their local model with the global model. The algorithm iterates local to a device, thus ensuring preservation of privacy. In addition, FL reduces the network load considerably by sending machine learning models instead of the training data. FL is commonly used in applications involving mobile and edge devices and, as shown in figure 1, generally FL participants can be binned into two major types: edge devices and organizations. Organizational clients, which are used in cross-silo applications, typically have higher computation and network resources and benefit from private datasets or devices that produce data. In contrast, edge devices have lower computation power and network throughput, and also have a smaller amount of training data than organizational clients.

While FL reduces network load, thus solving many distributed-ML problems, it cannot remove the communication overhead as the majority of iteration time is spent for transferring models. As common sizes of ML models fall

978-1-6654-9524-0/22/$31.00 ©2022 IEEE
DOI 10.1109/ICFEC54809.2022.00015

between 1 MB and 1 GB, reducing the time spent transferring models is of utmost importance [3] [4] [5]. Due to the varying resources available on clients [6], FL performance is bounded. Specifically, clients with poor resources (i.e. stragglers) can leave the server idle for long periods of time. Moreover, as the number of participating clients in FL can be in the order of millions, joining and leaving the client pool at any time during the training can result in server's network interface overflow and long aggregation time. Therefore, this makes the client sampling inevitable. In FL, Client Sampling tries to find a subset of clients which are more important and efficient for training the global model. After the sampling, ML models of the selected clients will be brought to the server for aggregation, further decreasing the network load. This also can make the training faster by recognizing and removing stragglers from the training process.

Most client sampling-based approaches try to solve the sampling problem using application parameters (loss function, norm of weight updates, etc.) [7] [8] [9] or system parameters (computation performance, network performance, etc.) [10] [11]. However, most of them are not able to jointly consider the system parameters and the model quality since the client selection is done prior to training. In addition, most of them do not provide the updated global model (the aggregated model that is generated in the central server) to the clients that are not selected which may cause the server to miss a client with critical information but poor resources. Moreover, in each iteration, only the selected clients update their local models with the global model, causing the majority of clients to work with older models. This decreases the chance of a device's selection in future iterations, thus widening the gap between its model and the updated model.

Here, we propose a client sampling protocol for FL which takes into account several parameters for selection such as accuracy and network throughput. The protocol allows every client to participate in each iteration of training in order to increase the accuracy of the global model and decrease the overall training time. The main contributions of this paper are listed as follows:

- We develop and implement a real-time FL protocol which utilizes a grading system for selecting the efficient clients. The protocol considers several system and application-related metrics, so we identify the parameters that have higher impacts on the final results and include them in the model. To guarantee fairness in participation, the majority of clients are surveyed by our ranking system after model training.
- We study the importance of different selection metrics by testing different configurations in the algorithm and show their trends throughout the training process.
- We investigate the impact of identically distributed data (IID) and non-IID training data on the proposed approach.

To evaluate our design, we implement and test our approach and compare it to the standard FL [2] and client selection scheme proposed in [10] which we refer to as FedCS. The rest of the paper is organized as follows. Section II contains background and related Work. Section III introduces the Intelligent Client Sampling protocol. Section IV describes the evaluation methodology, and section V and VI discuss results and conclusion.

II. Background and Related Work

The ever- increasing size of ML models and associated training data has identified scalability problems with traditional ML approaches. As a result, decentralized and distributed ML models have gained popularity owing to their ability to decompose tasks and data. Distributed ML approaches, such as [1], use a large amount of data to train deep ML models with billions of parameters by utilizing computation power of supercomputers. The server distributes data to multiple nodes, with each node individually training on the data it receives. A final model is then derived by aggregating the models trained on individual clients. In contrast, decentralized approaches such as FL [2] and Federated Forests, proposed in [12], benefit from heterogeneous edge and mobile devices and ensure data privacy by keeping the data inside each device and only transferring the ML models. Model transfer supports preservation of privacy, thereby enabling multiple organizations and edge devices to participate in training.

However, the size of ML models used in FL cause network saturation to occur as the number of participating devices scales up. In order to tackle the communication overheads in FL, compression techniques are utilized to decrease the size of data that is transferred. For example, [4] proposes a compression scheme which uses a quantization technique followed by encoding to make parallel stochastic gradient descent more efficient. Although, this reduces the communication load, the algorithm requires more iterations resulting in longer time to reach the desired accuracy. The works proposed in [3] and [5], use ternary quantization and sparsification methods in SGD to further compress the updates. Compared to the compression technique proposed in [4], these protocols compress the gradients in both upload download stages and are robust to Non-IID data.

To further decrease the network load, the number of clients participating can be reduced. Originally proposed in [2], FL distributes the global model to a random sub-sample of clients and only asks those clients to upload their ML models. This original approach can be improved using efficient client sampling based on different metrics with the goal of making the training process faster. The client sampling protocol introduced in [13] proposes a dynamic sampling scheme where the number of sampled clients is different in each iteration and shows that dynamic sampling can accelerate the earlier stages of the training process by selecting a smaller fraction compared to the later stages. Proposed by Ribero [7] and Chen [8], optimal client sampling for communication efficiency includes only clients that have weight updates exceeding a threshold. The work proposed in [9] samples clients using the quality of input images as a metric for selection. To ensure the inclusivity of sampling, [11] minimizes average model exchange time by considering client communication status while guaranteeing long-term fairness. In [14], first, the sampler calculates the Non-IID degree of data by observing

the defined weight divergence. The sampler then considers the Non-IID degree as the selection parameter and selects lower degrees more frequently.

Most of the FL applications are deployed in heterogeneous mobile device environments where the amount of available computation and network resources are varied based on the client type [15]. Therefore, client samplers with network and resource awareness must be developed. Nishio [10] proposes a FL protocol which estimates the completion (model downloading, training and uploading) time of the clients based on their resources and then distributes the global model only to the top ones. Then after training, it only includes the clients that were are able to finish before the deadline (some may still miss the deadline due to the stochasticity in the training and transfer time). However, this approach has limited inclusivity and may miss critical information from the clients with poor resources since it does not have any information about the quality of trained models.

The other common heterogeneity that exists in FL environments and applications is the imbalance in the distribution of data that results in non-uniform distribution of labels for a client. This can be solved by sampling the training data. As a continuation to Nishio's [10] client sampling protocol, in [16], authors propose a method which selects both ML models from clients and training data from the users that are allowed to share in order to make the algorithm efficient using Non-IID data. The FL approach proposed by Wang [17] first recognizes the amount of data heterogeneity for each client and then designs a loss function used for training. This loss function gives more importance to the weight updates of the labels with less population in the training data thereby mitigating the imbalance weight update effect caused by the training set.

III. Intelligent Client Sampling in Federated Learning

A. Client Selection Metrics

Most of the client selection parameters can be grouped into two major categories: system and application related. System related parameters represent the computation power and network status, while application parameters represent the quality of the trained model. Here, the model can be any model that tries to learn or to calculate an output in a federated scheme such as an averaging application in which the number of samples can show the dominance of a client. Latency to the server, network throughput, channel error rate and processor performance are possible systems parameters. However, the following can be grouped as application parameters: number of training data samples, distribution homogeneity, degree of non-IID, loss, accuracy, quality of training data (used in [9]), norm of gradients ([7] and [8]). As one of the goals of this work is to evaluate the importance of different metrics of selection, we will briefly go over the mentioned parameters and compare them.

As iSample aims to involve all clients in the training process, they are all provided with aggregated ML models from the server and are allowed to train their models. However, some clients may have poor computation resources and be late

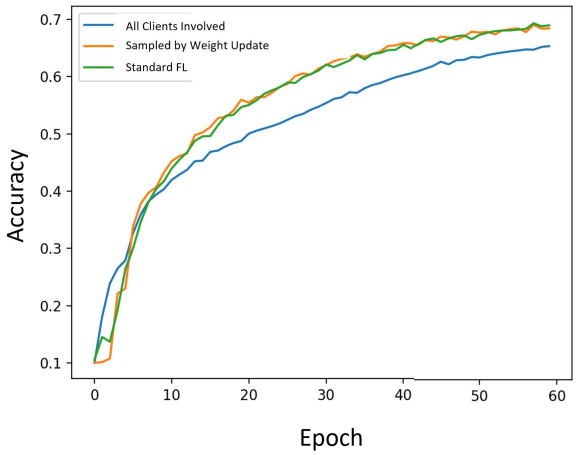

Fig. 2: Performance comparison of different FL approaches shows that sampling the clients by itself increases the accuracy.

to report the requested parameters to the server. In this case, the client can skip the current iteration and join in the next iterations' parameter reportings. In this study, we limit client selection parameters to only network-related parameters. This eliminates computation-based stragglers, allowing us to focus on network overhead.

From the network perspective, in iSample, two types of data are transferred through the network: ML models which are on the order of thousands to millions of parameters and selection parameters which are on the order of ten. Due to the size disparity, transfer time of an ML model heavily depends on the client's network throughput, while the transfer time of selection parameters is approximately equal to network latency.

When considering application-related selection parameters, the most feasible ones are data quality, size and distribution, all of which are available prior to training. For example, an increase in the degree of non-IID data will cause the model to have a lower accuracy during inference [14]. Although these metrics are commonly used in prior works, they do not show the performance of the model after training as precise as accuracy. Specially in a classification application where choosing based on loss does not always guarantee better performance and also can cause model overfitting. The accuracy in our application, which is an image classification for CIFAR-10 [18], is measured using test data which is homogeneous across all ten labels. However, in a non-IID setting, accuracy can be misleading because it does not show the model performance from each label individually and may not include clients with small amount but critical training data (for example, a rarely-encountered label). This issue is neglected in this approach since the protocol updates the local models of the majority of clients frequently but normalized accuracy with respect to each label can be a better metric compared to the general accuracy.

To evaluate the impact of parameters used in [7] and [8], we performed a test to compare with client selection based on weight updates with the original FL proposed in [2] and the

case all clients being involved in aggregation. The metric that is used for selection can be expressed as equation 1 where Δw_i is the change in value for a weight after one iteration of training. The test was done without considering network effects to purely observe its effect on accuracy. The ML model that was used is a convolutional neural network (CNN) with 122,570 trainable parameters (with the same configurations described in section IV) and the training data for each client was a random subsample of CIFAR-10 with 2000 samples and in each iteration, 5 clients were selected out of 30.

$$\|\Delta W\| = \sqrt{\sum_{i=1}^{K} \Delta w_i^2}, \tag{1}$$

where K is the number of weights in the model.

Shown in Figure 2, comparing the case that all of the clients are involved in the aggregation, both random and parameter-based selection increase the performance of the ML model. However, the accuracy of random selection and selection based on weight updates are almost the same throughout the training iterations. The other point to notice is that sampling by itself has increased the accuracy which can undermine some of the performance enhancements in client sampling.

For FL protocols, training performance is defined by the quality of the trained model and the time that is consumed to train the model. The quality of the global model is determined by the quality of the selected models that are participating in the aggregation. On the other hand, the time that is consumed to do each iteration of FL is determined by the slowest client (slow training and network communication) that participates in the aggregation. As our protocol, shown in figure 3, surveys the clients after the training phase, it is only affected by the model upload time which is directly related to the client's throughput (higher throughput results in less transmission time). Accordingly, both latency and network throughput are included in the grading system but the importance of throughput is significantly higher. Therefore, a grading system is required that can jointly consider the quality of models and their network condition.

$$grade = a.accuracy + b.throughput + c.\|\Delta W\| - d.latency \tag{2}$$

Equation 2 sets a grade for every client in each iteration of training in order to determine its efficiency for the overall training process based on four parameters with different types. Each parameter is normalized by the maximum amount that was reported in each client to make it dimensionless and the importance of each parameter is defined by its coefficient. The optimal set of coefficients (also here referred as configuration) for the equation can be different for each setup (type and size of the ML model) and iteration through the training process.

B. iSample Protocol

Providing inclusivity and efficiency for training in FL, while working in a heterogeneous environment shown in figure 1, requires a design of a protocol capable of supporting various devices ranging from low performance edge devices to organizations who benefit from highly-performant resources. The goal of this protocol is to select the most efficient clients based on the proposed grading system described in equation 2. In addition, the protocol provides an opportunity for all of the clients to be included in the aggregation phase regardless of their network and computation resources. As shown in Figure 3, in standard FL and FedCS, the model is distributed to a group of clients. Most of the work proposed to select efficient clients in FL evaluate the clients jointly considering computation and network conditions and remove the stragglers. However, looking from the server perspective in synchronous FL, the quality of the model (e.g. accuracy and loss) and the upload throughput of the client are important and determine the final performance.

Algorithm 1 iSample Server

1: **while** $Count < 0.8N$ in parallel **do**
2: Receive Parameters ($accuracy_n$, $throughput_n$, $\|\Delta W\|_n$, $latency_n$)
3: Calculate g_n using Equation (2) and store in G
4: $Count \leftarrow Count + 1$
5: **end while**
6: Sort G
7: Send Rq to the top $fraction$ of G
8: **while** Received $(0.8N)$ in parallel **do**
9: Receive(M_{ln})
10: **end while**
11: $M_g \leftarrow Aggregate(M_{ln})$
12: Send M_g to interacted clients
13: $Count = 0$
14: go to line 1

Algorithm 2 iSample Client

1: Initialize M_l
2: Train M_l for one iteration
3: Calculate Parameters ($accuracy_n$, $throughput_n$, $\|\Delta W\|_n$, $latency_n$)
4: **while** Parameters Sent **do**
5: Send($accuracy_n$, $throughput_n$, $\|\Delta W\|_n$, $latency_n$)
6: **end while**
7: Receive(Rq)
8: **if** $Rq = 1$ **then**
9: Send(M_l)
10: **end if**
11: Receive(M_g)
12: $M_l \leftarrow M_g$
13: go to line 2

The server proposed in our protocol assumes that N clients are provided with the initial models and they can train with their local data. Therefore as demonstrated in algorithm 1, the server starts by receiving parameters from the clients and proceeds to the next stage when it receives information from the majority of them. In the next step, the server calculates the grade(g_n) of each client using equation 2 and ranks them. The selected clients are requested (Rq) to upload their models

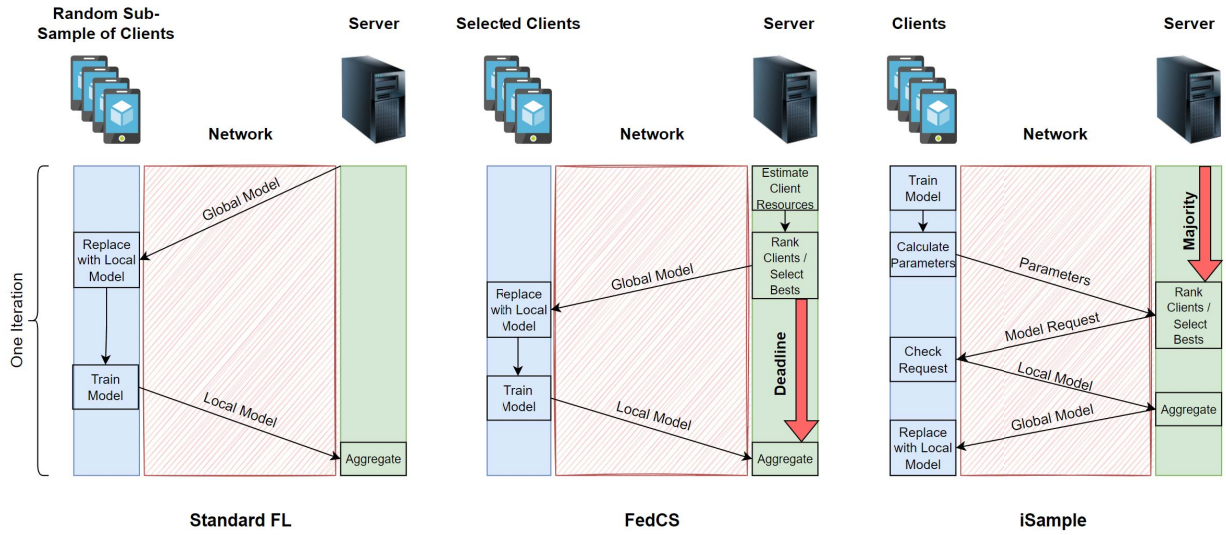

Fig. 3: Comparison of iSample protocol with Standard FL and FedCS

(Local Model: M_l) to the server for aggregation. Finally, the aggregated model (Global Model: M_g) for the next iteration of training, will be sent back to the clients that were able to deliver the parameters.

The remaining clients that were not involved in the first iteration wait at the send parameter stage shown in algorithm 2 line 5. As a result, they arrive first in the next iteration's parameter checking phase. This hides the inability of the clients who are slower than others at the expense of slow clients being surveyed less frequently. Additionally, the client models are updated as the server sends the aggregated model to all of the clients who participated in parameter checking. Accordingly, compared to other approaches, synchronization is maintained in the parameter receiving phase rather than aggregation or client selection.

IV. Evaluation Methodology

To evaluate our proposed protocol, we chose the Amazon Web Services (AWS) platform to implement and test the performance of our application. Server and clients are deployed using ML specialized Amazon EC-2 instances. In order to have a variety of clients, t2.small and t2.medium instance types were used which are different in computation power, as shown in table 1. The multi-threaded server was implemented on a t2.xlarge instance type to be able to handle multiple connections at the same time.

TABLE I: Computation and network resources of AWS EC2 T2 instance types [19]

Name	vCPUs	RAM (GiB)	Network Performance
T2.small	1	2.0	Low to Moderate
T2.medium	2	4.0	Low to Moderate
T2.xlarge	4	16.0	Moderate

To emulate a real FL application scenario, where clients are placed in different locations, EC-2 instances were launched in four different regions: Oregon, Northern Virginia, London and Frankfurt. The server was deployed in Ohio region which gave us the following communication latencies from the other four regions: 43 ms, 6 ms, 50 ms, 247 ms. Here, latency is the time it takes a data packet to travel from a client to server (the reported numbers are median of real measurements on the cloud). In addition, to imitate the network behavior of edge devices, the upload throughput of the EC-2 instances were limited to the range of 512 kbit/s to 2 Mbit/s uniformly using Linux Traffic Control Package [20].

The ML problem that was tested on the described system is an image classification application using training on CIFAR-10 [18]. The dataset has 50000 training images and 10000 test images which is homogeneously distributed among 10 labels. For an IID setting, training data for each client is a random subsample of the original dataset containing 2000 images. To evaluate the robustness of protocol in non-IID settings, 80 percent of the images are chosen from a single label and the rest are random samples from the training data. In every test, implementing Standard FL, FedCS or our protocol with different configurations, the training data is fixed and assigned to the dataset although random samples were chosen from the original dataset. The test data for evaluating both global and local models is the original test data with 10000 images.

The size of the transferred ML models directly affects the upload and download time between clients and the server. To show the impact of different coefficient configurations on different ML model sizes, two ML were tested. The first model is a CNN with 3 convolutional blocks, each including a 2D 3×3 convolution layer (each with 32, 32 and 64 channels) activated by ReLU and followed by a maxpooling layer. The model is concluded by a flatten and two dense layers at the end to predict the correct label resulting in 122570 trainable parameters which has the size of 0.46 MB. We refer to this model as CNN. The second model, which is referred as VGG, is a VGG [21] with three convolutional blocks containing two

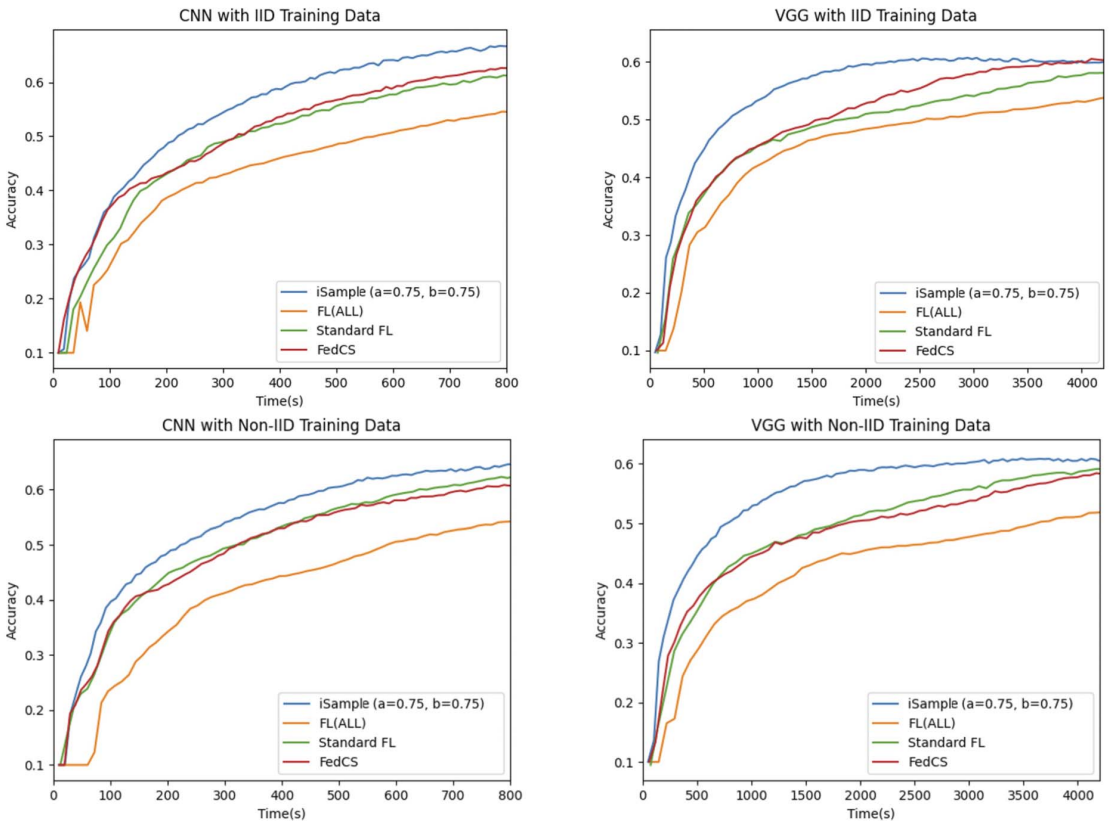

Fig. 4: Comparison of one configuration of iSample with FedCS and Standard FL using different ML models for IID and non-IID training data

back-to-back 2D 3×3 convolution layers (each with 32, 32, 64, 64, 128 and 128 channels). The tested VGG model has 1060130 trainable parameters which results in 4.04 MB size. In each iteration of all of the tested approaches, ML models are trained for one epoch and then aggregated. The optimizer used for training was Adam with a learning rate of 0.001 .

The approach proposed by Nishio [10] was selected as a baseline for comparison to our work in addition to the standard FL and referred to as FedCS. In order to implement FedCS [10], as shown in figure 3 , in the described environment, the T_{round} parameter was needed which is the server deadline to receive the ML models. However, the network condition and the CPU performance in our setup is different than the simulation environment used in [10] to test FedCS. To deal with this discrepancy, the deadline was calculated as the time needed for an average client to download, train and upload its model.

All of the protocols iterated for 100 rounds. There were 80 clients in total and in each round of Standard FL and iSample, 16 were selected, randomly and by the grading system respectively. For FedCS, the global model was distributed to half of the clients in each round and then sampled based on the deadline, therefore the number of sampled clients was different. Also, the case that all of the clients are involved in the aggregation is tested and referred to as FL(ALL) in the results section.

As the effect of weight update (shown in figure 2) was not striking, to narrow the search space of coefficients, the parameter was included in the grading system but its coefficient was set to 0.1 ($c = 0.1$). Also, latency's coefficient was set to 0.01 ($d = 0.01$). Our experimental evaluation tries to show how different coefficient selections affect the final performances of trained models. In addition, to show the effect of sampling without jointly considering accuracy and throughput, four extreme configurations are chosen. The configurations that only considers throughput (and almost neglects the accuracy) are expected to always select the clients with the best network resources even if they have significantly lower accuracy. Therefore, they may have better accuracy at the very early stages of training but will encounter a ceiling, especially in non-IID settings, since it always selects the same clients. On the other hand, the configurations that only consider model quality can achieve better accuracy at the end of training but will not be efficient because they prolong the iteration time by waiting for an accurate but slow client.

V. Results

The performance of different approaches for equal runtimes are shown in Figure 4. To have a fair comparison, we used the following coefficient comparison to promote fairness: $a = b = 0.75$. In all of the tested scenarios, FL(ALL) had

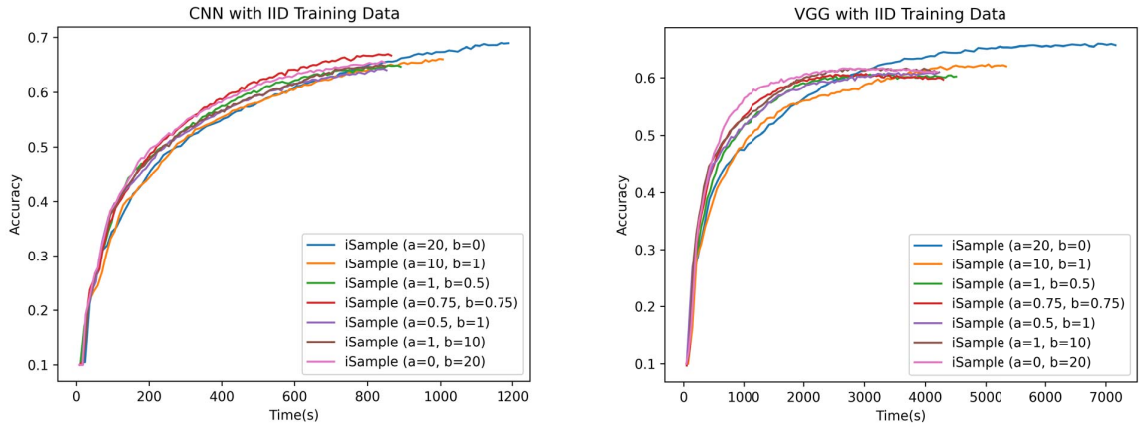

Fig. 5: Performance comparison of different coefficient configurations (100 iterations)

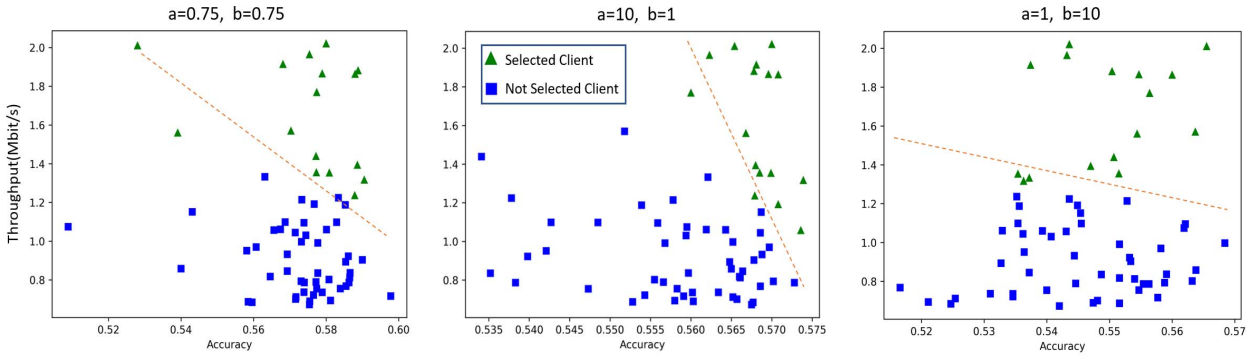

Fig. 6: Selected clients for CNN, training with non-IID data at epoch 50 using iSample with different configurations

the lowest performance. This is due to the sampling effect (discussed in section III.B) decreasing accuracy. Additionally, the heterogeneity of clients (as in FL(ALL) reduces runtime performance. The results show that iSample outperforms Standard FL and FedCS especially in the earlier stages of training. iSample achieves higher accuracy in a shorter time because it considers the network condition of the clients and can finish more iterations in an equal runtime. In addition, iSample improves the accuracy by considering the clients' ML model quality since it receives the parameters from the clients after training. Moreover, as shown in figure 4, iSample, as was Standard FL (due to the randomness in selection), is robust to non-IID settings. FedCS was inferior as never selects the clients with poor resources even if they have critical information.

Comparing VGG to CNN, iSample still outperforms the other baselines in the earlier stages of training. However, when the models get saturated, iSample ends up having a higher accuracy by a smaller margin. This can be explained using figure 5, where the figure shows how different coefficient configurations of iSample can change the performance of the protocol. The best configuration for CNN is the red curve which has the following values: $a = b = 0.75$. However for VGG, in the early stages, the best configuration is the pink curve ($a = 0, b = 20$) up until the saturation point (around time 3000s) where the blue curve ($a = 20, b = 0$) starts to take the lead (same as CNN). Size of the VGG network is larger than CNN by 10 times therefore, it consumes most of the training iteration time to transfer the VGG model. As a result, for VGG, the importance of the throughput (pink curve) is much higher than accuracy comparing to CNN. Moreover, it can be observed from figure 5 that after the saturation point, the importance shifts from throughput toward the accuracy and the blue curve can improve the accuracy of the model further. This gives us the idea that for the larger ML models, the coefficient set can be changed dynamically through the training process.

In addition, although iSample is tested in an environment emulating mobile and edge devices for FL, the protocol can be utilized to enhance the performance of distributed ML applications as well. Distributed ML applications are trained using clusters where nodes may have heterogeneity in network resources due to the non-uniform memory access (NUMA) effect. Therefore, iSample can make the client sampling in clusters more efficient.

In figure 6, the distribution of clients by accuracy and throughput for different configurations at the same iteration can be observed. The figure shows the majority of clients out of 80 that were able to deliver their parameters to the server, indicating that a client with the highest accuracy will always be

selected. In addition, as there are two other parameters in the grading system, a few of the clients are selected from the other side of the lines. Another point to note here is that although iSample increases the performance of global model, it is not as energy efficient as other approaches that were discussed. This is due to the fact that iSample requires the majority of the clients to do the training. As figure 6 shows, in this iteration, 64 clients trained their models and only 16 were selected, where as in the Standard FL, only 16 clients have trained their models.

Finally, table 1 and 2 show the average and variance of the iteration times for different tested approaches. The selected configuration of iSample for comparison($a = b = 0.75$), while using CNN, improves the iteration time by 27% comparing to Standard FL and by 18% comparing to FedCS. On the other hand, for VGG, iSample reduces the iteration time by 39% comparing to Standard FL and by 29% comparing to FedCS.

TABLE II: Mean and variance of iteration time using CNN

Mean, Variance	IID	Non-IID
iSample(a=1, b=0.5)	8.90, 0.150	8.85, 0.139
iSample(a=b=0.75)	8.64, 0.051	8.66, 0.087
iSample(a=0.5, b=1)	8.51, 0.016	8.49, 0.012
Standard FL	11.86, 0.037	11.88, 0.039
FL(ALL)	11.99, 0.001	11.98, 0.001
FedCS	10.58, 0.007	10.57, 0.012

TABLE III: Mean and variance of iteration time for VGG

Mean, Variance	IID	Non-IID
iSample(a=1, b=0.5)	45.06, 14.50	44.70, 8.634
iSample(a=b=0.75)	42.93, 3.094	42.88, 3.137
iSample(a=0.5, b=1)	42.27, 1.622	42.15, 1.142
Standard FL	71.23, 4.451	71.61, 2.767
FL(ALL)	73.25, 0.003	73.22, 0.003
FedCS	60.47, 0.533	60.36, 0.854

VI. Conclusion

Although FL reduces the communication load of decentralized learning approaches, the heterogeneity in mobile and edge devices environment can constrain the training performance. Therefore, client selection techniques are proposed to select the efficient clients for training. In most of the current approaches, the client selection is done prior to model training which makes the server unable to consider the quality of the trained models. Additionally, current solutions provide the aggregated model only to a subsample of clients thus undermining the inclusivity. We proposed a new FL client selection protocol, iSample, with the goal of training time minimization and performance enhancement to address the aforementioned issues. The protocol utilizes a grading system for recognizing efficient clients and allowing them to participate in aggregation. As iSample surveys the majority of clients after model training, it is able to maintain inclusivity. It was shown that iSample achieves robustness to non-IID settings by considering model quality and updating the client models more frequently than state of the art solutions. The protocol was implemented and tested on the AWS platform and compared with FedCS and Standard FL. The results show that iSample has decreased the training time for both CNN and VGG by 27% and 39% comparing to Standard FL. Future work involves adding more parameters to the grading system and employing reinforcement learning techniques to change the protocol parameters dynamically during the training process.

Acknowledgment

This work is funded by the NSF IDIEA-DC program as award number 2038682 under the NSF Office of Advanced Cyberinfrastructure (OAC) organization.

References

[1] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker *et al.*, "Large scale distributed deep networks," 2012.

[2] 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*. PMLR, 2017, pp. 1273–1282.

[3] J. Xu, W. Du, Y. Jin, W. He, and R. Cheng, "Ternary compression for communication-efficient federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[4] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," 2017.

[5] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[6] A. Jahanshahi, R. Sharifi, M. Rezvani, and H. Zamani, "Inf4edge: Automatic resource-aware generation of energy-efficient cnn inference accelerator for edge embedded fpgas," in *2021 12th International Green and Sustainable Computing Conference (IGSC)*, 2021, pp. 1–8.

[7] M. Ribero and H. Vikalo, "Communication-efficient federated learning via optimal client sampling," *arXiv preprint arXiv:2007.15197*, 2020.

[8] W. Chen, S. Horvath, and P. Richtarik, "Optimal client sampling for federated learning," *arXiv preprint arXiv:2010.13723*, 2020.

[9] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020.

[10] 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)*. IEEE, 2019, pp. 1–7.

[11] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Y. Zomaya, "An efficiency-boosting client selection scheme for federated learning with fairness guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1552–1564, 2020.

[12] Y. Liu, Y. Liu, Z. Liu, Y. Liang, C. Meng, J. Zhang, and Y. Zheng, "Federated forest," *IEEE Transactions on Big Data*, 2020.

[13] S. Ji, W. Jiang, A. Walid, and X. Li, "Dynamic sampling and selective masking for communication-efficient federated learning," *arXiv preprint arXiv:2003.09603*, 2020.

[14] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-iid data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24 462–24 474, 2021.

[15] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 46–51, 2020.

[16] 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.

[17] L. Wang, S. Xu, X. Wang, and Q. Zhu, "Addressing class imbalance in federated learning," 2020.

[18] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "Cifar10-dvs: An event-stream dataset for object classification," *Frontiers in Neuroscience*, vol. 11, no. 309, May 2017.

[19] "Amazon EC2 T2 Instances," https://aws.amazon.com/ec2/instance-types/, Online; Accessed November 10, 2021.

[20] W. Almesberger, "Linux network traffic control – implementation overview," 1999.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.