FedLR: A Federated Learning based Approach for Faster Communication in Edge Environment

Vipul Singh Negi
Department of Computer Science and
Engineering
National Institute of Technology
Rourkela, India 769008
vipulhld001@gmail.com

Suchismita Chinara
Department of Computer Science and
Engineering
National Institute of Technology
Rourkela, India 769008
suchi.nitrkl@gmail.com

Abstract—Data are key to the survival of any modern data scenario. With the rise of IoT and Edge Computing, millions of data are shared in a second. The requirement for sound distributed systems is high. One of the best systems is Federated Learning, which works by sharing training parameters rather than training data. This makes it the perfect choice for distributive machine learning. FedAvg is the vanilla algorithm for Federated Learning, but it has a glaring problem of not working correctly for slacker devices. A new approach, FedLR, is proposed in this manuscript to remedy the problem of slacker devices. It provides all the benefits of FedAvg plus it reduces the communication costs as well as creates a fair scenario by also using the slacker devices. The proposed work is compared with FedAvg and FedProx. FedProx is one of the federated learning systems that works for heterogeneous clients while preserving slacker devices. The challenge was to make a fair and communication-efficient strategy for federated learning. It was achieved using FedLR, which uses a time variable to classify devices as slackers and tunes them to handle less data in order to make them contribute to the federation and keep it fair.

Index Terms—Federated learning, IoT, Deep Learning, Edge Computing

I. INTRODUCTION

Federated Learning is a new computing technology that promises to provide all the benefits of AI and ML while preserving users' privacy and security. Instead of training on individual users' data, it trains on the client's weight, which is stored locally with the clients'. This system preserves the user's privacy and provides meaningful AI benefits for many applications of distributive computing.Example: WeBankAI [1], Nvidia Clara [2], and Google Keyboard [3]. That does not mean that it does not have any flaws. It is prone to man-in-the-middle attacks and data poisoning, to name a few. Federated Learning is very suitable for IoT and Edge devices as these are well connected to the internet and send vast amounts of data. The Number of connected IoT devices has grown by 13% in the year 2024, which leads to 18.8 billion global devices [4].

Federated Learning provides privacy, scalability, cost-effectiveness, and efficiency. However, each of these features comes at a cost, especially in terms of efficiency. The Federated Learning process is as efficient as its slowest device (slacker), so if someone with older hardware connects to the federation (Fig 1.), the whole system will be bogged down,

and the classic FedAvg [5] will remove that device from training. So, one of the biggest problems identified is the slacker devices.
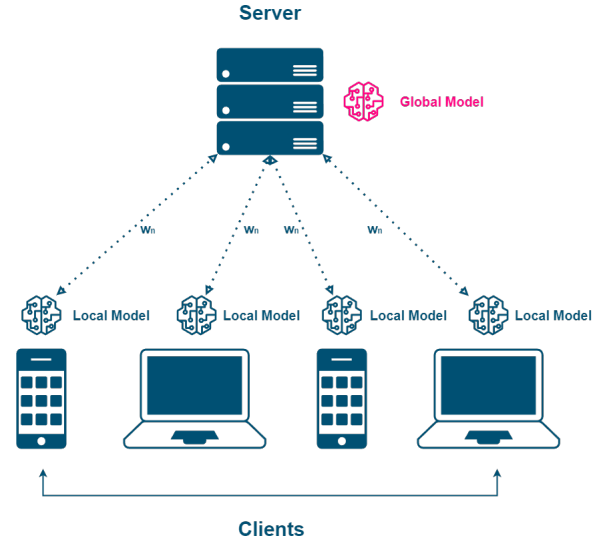

Fig. 1. Workings of Federated Learning.

In this paper, we present a new strategy which significantly addresses the problem with the old one [5], keeping the effectiveness of federated learning alive. The problem with the vanilla FedAvg is the client selection methodology, which is taken at random, and the device's participation has no threshold value to meet in order to participate in the federated learning process. Another problem with FedAvg is dealing with device heterogeneity, as each device usually tends to have a different CPU, RAM, and GPU, which are the essential blocks of any neural network. FedProx [6] remedies the heterogeneity of FedAvg to a certain point but doesn't account for fairness. Fairness means all clients should be treated as fair, and their contributions to the global model should carry equal weight. The proposed strategy has been named FedLR, which is a dynamic learning rate and epoch-based client selection strategy.

The proposed work focuses on reducing the total commu-

nication time taken by federated learning. Chen et al. [7] used RAM and CPU utilization data from clients in order to boost client selection. But getting such data from clients could lead to device fingerprinting [8] [9], which puts federated learning at risk. In this paper, we are employing FedLR, a learning rate based on efficient communication; by considering heterogeneous clients, a time variable has been calculated, and according to that time variable, qualitative decisions towards learning rate selection have been made. In the proposed methodology, a time variable has been calculated, and according to that time variable, qualitative decisions towards learning rate selection have been made. FedLR guarantees that by considering heterogeneous clients, a time variable has been calculated, and according to that time variable, qualitative decisions towards learning rate selection have been made faster communication time with a fair selection of clients as it always considers slacker devices, which makes this methodology Fair. The main contributions of the proposed work are summarized as follows:

1) By conducting extensive sets of experiments we came to a quantitative conclusion about the effects of learning rate (η) in federated learning which is highly effective parameter in federated learning.
2) The next contribution is regarding individual clients epochs. The time variable, which represents each client's runtime, is used to find out the slacker devices and adjust the epoch values to reduce the number of slacker devices.
3) Pairing the time variable with η and epoch values of client to perform better client selection which focused on fairness and heterogeneity. A client selection strategy FedLR has been proposed.

The ultimate goal of this work is to present a federated learning approach which reduces the communication cost as well as preserves the Accuracy of the client's model. The remainder of the paper is organized as follows: Section II discusses Related Works. Section III explains the problem statement of the literature. Section IV introduces FedLR and explains its key features and how it performs them. Section V does the result and analysis of the proposed algorithm, and Section VI finally concludes this paper.

II. RELATED WORKS

The idea of privacy-preserving deep learning was first presented by Shokri et al. [10] using distributed and selective SGD to make deep learning models privacy-preserving. After this the Federated learning terminology was coined by McMahan et al. [5] in which they proposed the Federate Averaging algorithm and tested it using the MNIST Digit Dataset. The data was partitioned into IID and Non-IID. In Non-IID data, only two-digit data were given to the clients. They have also used CIFAR-10 in the balanced and IID settings.

Li et al. [6] proposed an update to the traditional FedAvg called FedProx by adding statistical heterogeneity (Proximal Term), which handles client heterogeneity better than the previously proposed method.

Zhang et al. [11] has proposed AdaptFl which is an adaptive FL framework specific to heterogeneous devices. It works on taking the GPU, CPU, Battery, and Network information of the clients. AdaptFl has proposed formulas for calculating energy, transmission, and utilization efficiency. But has the same problem as Chen et al. [7] because taking this much client information will make it vulnerable to device fingerprinting.

Zhang et al. [12] have proposed FedSL, which uses split layer aggregation and divides the global feature extractor into smaller groups. Then the clients are scored and concatenated back to the original state. [12] have considered [5] and [6] as using 100% of communication cost.

Ma et al. [13] has proposed an adaptive batch sizes FL approach for resource-constrained devices. The test bed uses one workstation as a server and 10 NVIDIA Jetson's as clients. Orlandi et al. [14] has proposed an entropy-based FL for Edge intelligence environments and focuses on non-IID data. 19 clients are created using VMs and have been each provided with 4GB of RAM. It calculates entropy and then compares the global mean entropy for each client. The data with lower entropy will be selected for local training; otherwise, it will be discarded.

Mills et al. [15] have proposed having a decaying number of local SGD steps, which makes the FL system more efficient. This is also one of the inspirations for the proposed FedLR approach. Mathur et al. [16] has implemented federated learning using the Flower framework. They have implemented federated learning in 5 mobile devices (three phones and two tablets). They have used the ever-popular MobileNetV2 [17] architecture. Negi et al. [18] has also studied the same [17] architecture for federated learning for FedAvg benchmarking.

Hu et al. [19] proposed a gradient similarity-based client selection algorithm. It prioritises the clients with similar gradients by calculating cosine similarity. This method will lead to the unfair selection of clients as it doesn't consider slacker devices. Ye et al. [20] have proposed Eco-FL, which is much more sustainable than [5] and [6] in terms of energy consumption. [20] has also used the flower framework to implement the Eco-FL strategy. [21] and [22] are used as datasets, and they have tested Eco-Fl in both IID and Non-IID settings.

III. PROBLEM STATEMENT

The FedAvg proposed by [5] opened the doors for communication-efficient deep learning networks as they proposed the Federated Averaging methods to train models collaboratively without sharing the underlying training data to preserve the privacy, and ever since then, it has become the state of the art method for Privacy-preserving Deep Learning Models.

A. Problem Setup

The vanilla FedAvg algorithm has some problems when it comes to client selection, heterogeneous computational resources, and fairness, as it focuses primarily on the privacy-preserving aspect of the learning. The algorithm begins with

initializing the global model at random or for pre-trained data. This begins the client selection process, which selects clients at random. The global weights are distributed to the clients and the training begins and only stops when we get our desired outcome. To remedy the heterogeneity of FedAvg, a new strategy, FedProx [6], was proposed, which tries to contain the dropping slacker devices due to the heterogeneous nature of the devices.

B. Client Selection

In the vanilla FedAvg, the clients are selected at random using

$$m \leftarrow max(C \cdot K, 1) \tag{1}$$

where C is the fraction of selected clients and K is the actual clients, which gives us the total number of m clients for the federated learning. This method of selecting clients is good, but it can easily be improved upon by proposing a new strategy for client selection by adding some more criteria for client selection. The proposed strategy should improve over the existing one by making the process more robust and highly communication efficient. A new parameter is required, which can perform better client selection than the current Equation 1.

C. Heterogeneous Computational Resources

The devices participating in federated learning are not always homogeneous. The chances of it being heterogeneous are higher. A survey done by Markets and Markets [23] states that 35 key players exist in the edge devices manufacturing phase, with some providing devices while other provides the processors. This many key players means that a major amount of devices are heterogeneous, which makes client selection much more complicated because as the diversity increases, the number of slacker devices also increases. The slowest performers Slacker devices are the devices which are the slowest performers in the system. The motto of federated learning is the faster the slacker device the model will be equally efficient. So that means in order to propose a better strategy, one must figure out a way to improve upon the slacker devices.

D. Fairness

Fairness is a unique thing that removes discrimination between clients and ensures that all the device's contributions are taken into consideration. The best way to understand this problem is to imagine two hospitals, H1 and H2, with H1 being a bigger hospital with various specializations and H2 being a small clinic. Communicable Diseases are the most popular ones and easy to get, and if H1 is getting patients for one such communicable disease, that means it is the same for H2 as well. However, due to the size of H1, the data gathered from H2 will be minuscule, but that doesn't mean that it's irrelevant. So, one must also consider fairness as one of the major components of creating a new strategy. Ezzeldin et al. [24] have proposed a Fairfed strategy to solve this problem using the debiasing method across clients.

IV. Proposed Algorithm: FedLR

After discussing the problem with the existing systems, FedLR, a learning rate and epoch-based strategy targeting slacker devices, is proposed. The proposed FedLR is broken into three segments: Learning Rate Selection, Epoch Selection for individual clients, and Client Selection based on the Client Runtime. The objective is to reduce the cost of efficient communication while preserving the accuracy of the model. A slacker device-centric algorithm has been proposed, focusing on the slowest devices and considering the fairness of client selection. A Federated Learning System is as fast as its slowest device. This is the basic principle of federated learning, so the proposed algorithm for client selection primarily focuses on optimizing the slowest devices in the federation. The algorithm is based on turning the learning rate and epochs of Individual clients. A time quantum has been proposed; the clients are classified as quicker or slacker based on the help of this time quantum. To validate if the learning rate-based assumption (Section IV A) is correct (the bigger and smaller lr are assigned for quicker and slacker devices, respectively). However, this will only reduce the total communication time by a few seconds, so a new epoch for the slacker devices is proposed below in equation 2. To make it effective (Section IV B).

A. Learning Rate Selection

The algorithm 1 below explains the learning rate (η) selection by the clients. The learning rates are taken with an increment of ten. After running different η, it was observed that the accuracy was similar, but the total communication times were being reduced with smaller η values. All three η cases were tested and they showed a good amount of change in communication cost. The dataset used for these experiments was the CIFAR10 dataset. The results are similar to the ones shown in [18].

Algorithm 1 Learning Rate Selection The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate

---
1: **ClientUpdate**(k, W):   ▷ Run on client k
2: $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
3: **for** each local epoch i from 1 to E **do**
4:      Fit η as [0.01, 0.001, 0.0001] from the Server.
5:      **for** batch $b \in \mathcal{B}$ **do**
6:          $w \leftarrow w - \eta \nabla l(w; b)$
7:      **end for**
8: **end for**
9: return w to server

---

1) Case 1: FedLr (lr =0.01) with 10 Rounds and 5 Epochs: The best accuracy we achieved in this scenario is 56.6%. This works similarly to the FedAvg Strategy and claims almost similar runtime the gap is 2s. The losses also perform similarly to the base FedAvg Strategy. Table I shows the result of the experiment.

2) Case 2: FedLr (lr =0.001) with 10 Rounds and 5 Epochs: The best accuracy is 56.2%. The runtime is reduced by 6 seconds, which makes it half a second per round. However, the losses are greater compared to Case 1. In Case 1, the lowest

FedLR (lr = 0.01) with 10 Round and 5 Epochs					
Clients	Accuracy	Loss	Local Acc	Local Loss	Total Time
1	56.2%	0.063	95.3%	0.0050	
2	52.4%	0.065	95.7%	0.0047	
3	51.2%	0.069	94.7%	0.0053	835.6s
4	**56.6%**	0.068	94.6%	0.0056	
5	52.4%	0.061	94.9%	0.0054	

loss was 0.0047; in Case 2, it was 0.0066. Table II shows the result of this experiment.

FedLR (lr = 0.001) with 10 Round and 5 Epochs					
Clients	Accuracy	Loss	Local Acc	Local Loss	Total Time
1	55.8%	0.065	93.4%	0.0066	
2	56%	0.059	93.7%	0.0064	
3	52.4%	0.069	93.5%	0.0067	830.9s
4	52.4%	0.069	93.7%	0.0065	
5	**56.2%**	0.059	93.4%	0.0066	

3) Case 3: FedLr (lr =0.0001) with 10 Rounds and 5 Epochs: The best accuracy is 57.6% with runtime reduced by 9 sec. While having similar losses as Cases 1 and 2. The full results are in table III below.

FedLR (lr = 0.0001) with 10 Round and 5 Epochs					
Clients	Accuracy	Loss	Local Acc	Local Loss	Total Time
1	54.8%	0.063	93%	0.0060	
2	**57.6%**	0.062	95%	0.0053	
3	56.8%	0.063	95.2%	0.0050	826.0s
4	54.4%	0.065	95.1%	0.0052	
5	53.4%	0.062	94.6%	0.0056	

B. Epoch Selection for Individual Clients

The first step was to figure out suitable η values which provides us with significant changes but just changing the lr values in not sufficient for us to make the strategy robust. The next step is to provide individual epoch values for each client based on a balanced parameter. Chen et al. [7] used the CPU and RAM metrics, which provide us with significant changes, but just changing the η values of clients in order to boost their strategy. This methodology is effective but is not fully privacy preserving. Device fingerprinting is a big risk to the Federated Learning scenario. Li et al. [6] proposed a Proximal term to improve the slacker devices which were dropping because of delays in computing E. The proposed equation 2 calculates local epochs for individual clients.

$$E_n = \left\lfloor \frac{mode[T_q] - count\_of\_value}{mode[T_q]} \right\rfloor * E \qquad (2)$$

In Equation 2. $mode[T_q]$ is the mode value from the time quantum, and *count_of_value* is the time the mode value is repeated. The equation 2 combined with the equation 1 for

client selection improves it significantly. The key values for the Algorithm 2 are as follows:

1) The K clients are indexed by k.
2) C is the fraction of the client selected.
3) B is the local minibatch size.
4) E is the number of local epochs.
5) η is the learning rate.
6) T_q is time quantum.
7) E_n is the new epoch.

Algorithm 2 FedLR The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs and E_n is the new epoch, T_q is the time quantum, and η is the learning rate

---

1: **Server Executes**
2: initialize w_0
3: **for** each round $t = 1, 2, ..$ **do**
4:     $m \leftarrow max(C \cdot K, 1)$
5:     $S_t \leftarrow$ (random set of m clients)
6:     **for** each client $k \in S_t$ **in parallel do**
7:         $w_{t+1}^k \leftarrow ClientUpdate(k, w_t)$
8:     **end for**
9:     $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
10: **end for**
11: **ClientUpdate**(k, W): ▷ Run on client k
12: $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
13: **for** each local epoch i from 1 to E | E_n **do**
14:     Find T_q (T_q is the epoch runtime of individual clients).
15:     Find mode[t_q]
16:     **if:** mode[t_q] is smallest
17:         i = E and η= Bigger Learning Rate
18:     **else:**
19:         Find E_n using equation 1.
20:         i = E_n and η = Smaller Learning Rate
21:     **for** batch $b \in \mathcal{B}$ **do**
22:         $w \leftarrow w - \eta\nabla l(w; b)$
23:     **end for**
24: **end for**
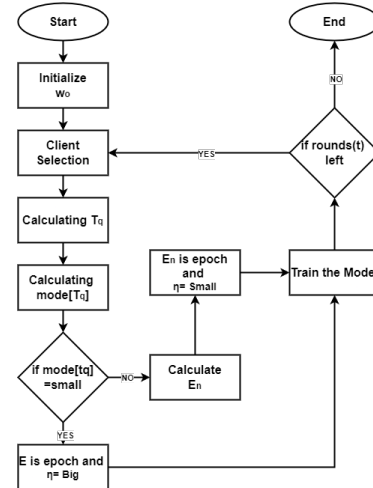25: return w to server

---

C. Working of FedLR

Fig. 2. Flow Chart describing the workings of FedFL.

Fig. 2 shows the workings of FedLR. Initialize the global parameter value w_0. Selection of all the eligible clients for training. Finding each client's run-time as they finish training.

Finding the mode time value by comparing the individual times. Selecting the learning rate and epoch values based on the mode value. The bigger learning rate (η) will be assigned to quicker devices with the default epoch (E) value. The slacker devices will be assigned a lower learning rate with a lower epoch value En using equation 1. and a smaller η. The time of these devices will be compared next, and the slacker devices will receive a low amount of work but will still be a pivotal element in the federated learning process.

V. RESULT AND ANALYSIS

The model architecture used in the experiments is a 2 layered Convolution Neural Network. The list of datasets used for the experiments are as follows:

1) CIFAR10 [21] dataset containing ten classes.
2) MNIST [25] dataset of Handwritten digits containing ten classes.
3) FMNIST [26] dataset of fashion items containing ten classes.
4) CIFAR100 [22] dataset containing hundred classes.

A. Machines Used

Two machines are used to test the proposed algorithm, and the specifications of the machines are given below:

1) Google Collab (Free Tier with T4 GPU).
2) Workstation (128GB RAM, Intel Xeon Silver 4216 CPU, and NVIDIA RTX A4000 GPU)

All clients use the full datasets and run at 10 Epochs and 10 Communication Rounds. Using .25 and 1 GPU and CPU, respectively in Flower. The Results of our Experiments are as follows: 10 Clients using FedAvg, FedProx, and FedLR on all Datasets. The blue dotted lines represent the local loss and accuracy, while the orange ones represent the global loss and accuracy. The legends show how many total epochs one client takes.
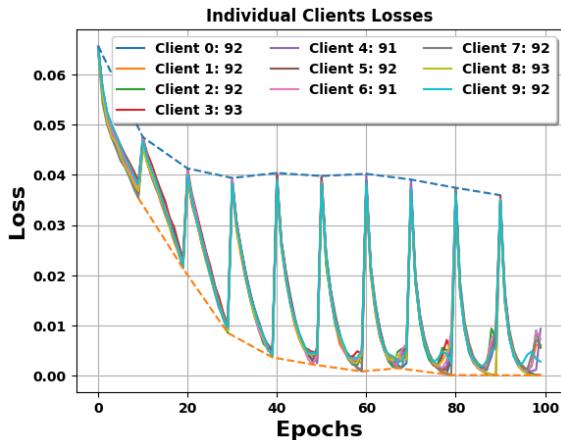

Fig. 3. Loss of 10 Clients using FedAvg on the CIFAR10 Dataset (Workstation).

Fig. 3, 4 and 5 represent the loss, accuracy, and Individual time taken by each client, respectively, for FedAvg. These are
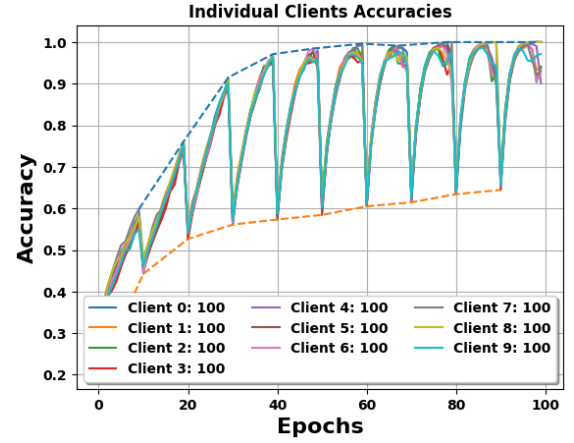

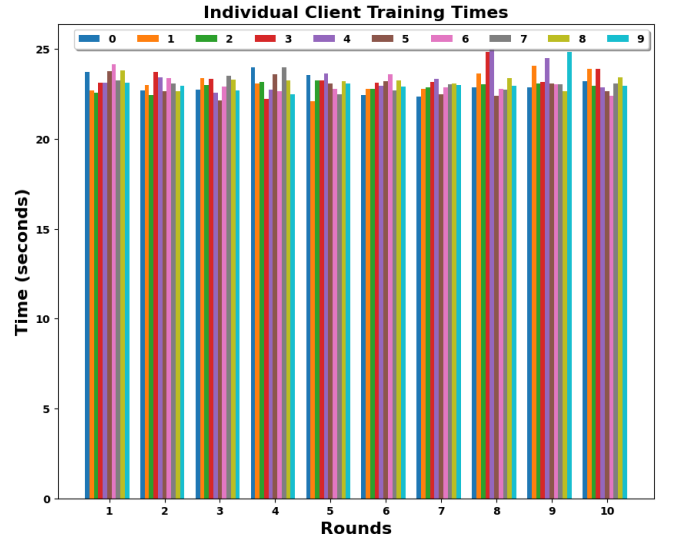Fig. 4. Accuracy of 10 Clients using FedAvg on the CIFAR10 Dataset (Workstation).

Fig. 5. Individual Time of 10 Clients using FedAvg on the CIFAR10 Dataset (Workstation).

workstation times and the accuracy and losses are similar to FedLR and FedProx. The blue dotted lines represent the local loss and accuracy, while the orange ones represent the global loss and accuracy. The legend of the fig. 3 shows how many total epochs one client takes.

Fig. 6, 7 and 8 represent the loss, accuracy, and Individual time taken by each client, respectively, for FedProx. FedProx performs a bit better in terms of total communication cost than FedAvg in some scenarios.

Fig. 9,10 and 11 represent the loss, accuracy, and Individual time taken by each client respectively for FedLR. In fig. 9, each client runs a different number of epochs. The ones with the lowest epochs are the Slacker devices; hence, they contributed less. It is to be noted that the resources of each client are given at random to promote heterogeneity. Hence, none of the clients completed 100 E due to them being slackers in a few rounds.
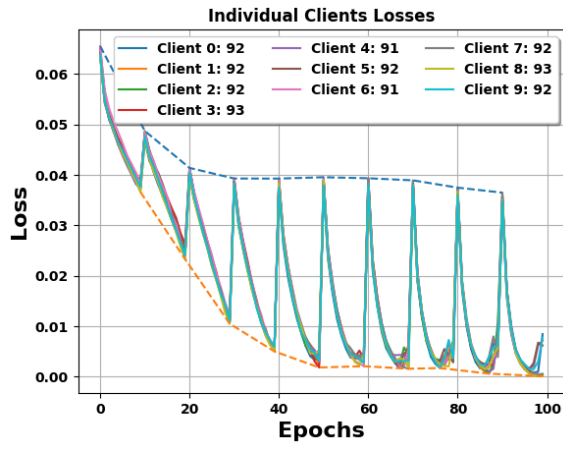
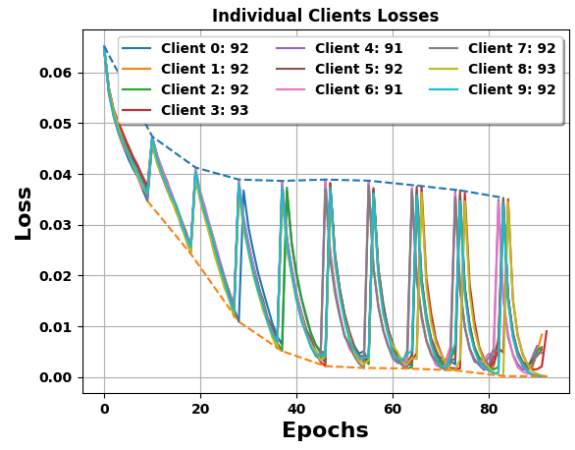Fig. 6. Loss of 10 Clients using FedProx on the CIFAR10 Dataset (Workstation).

Fig. 9. Loss of 10 Clients using FedLR on the CIFAR10 Dataset (Workstation).
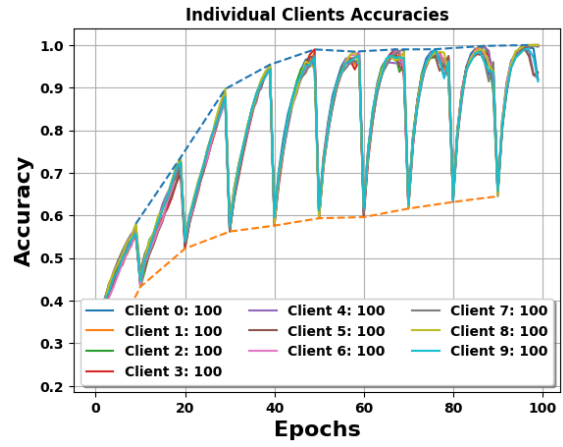

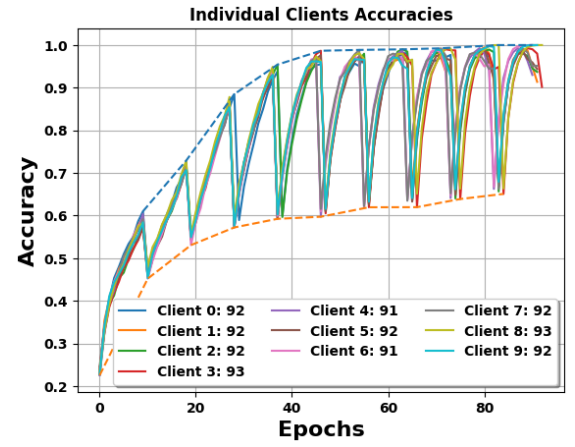Fig. 7. Accuracy of 10 Clients using FedProx on the CIFAR10 Dataset (Workstation).

Fig. 10. Accuracy of 10 Clients using FedLR on the CIFAR10 Dataset (Workstation).
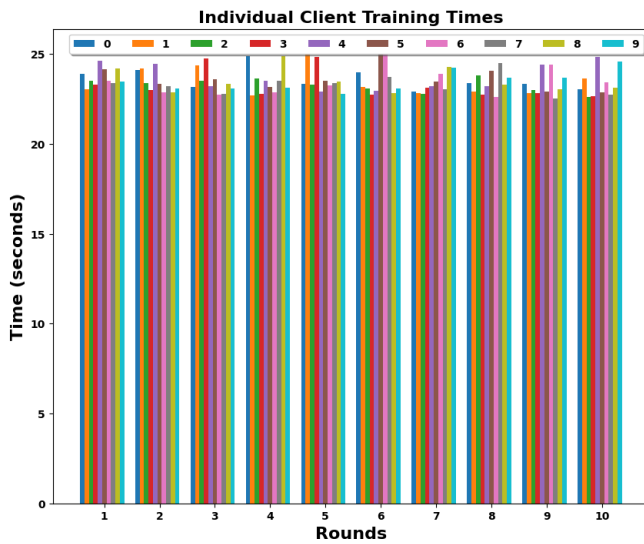

Fig. 8. Individual Time of 10 Clients using FedProx on the CIFAR10 Dataset (Workstation).
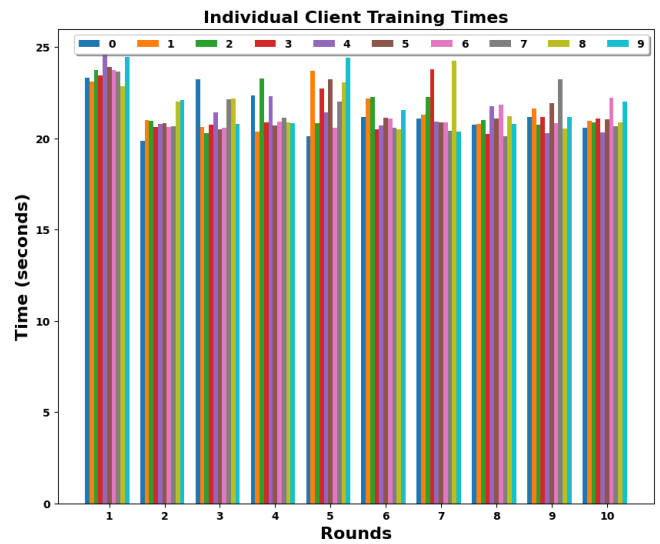

Fig. 11. Individual Time of 10 Clients using FedLR on the CIFAR10 Dataset (Workstation).
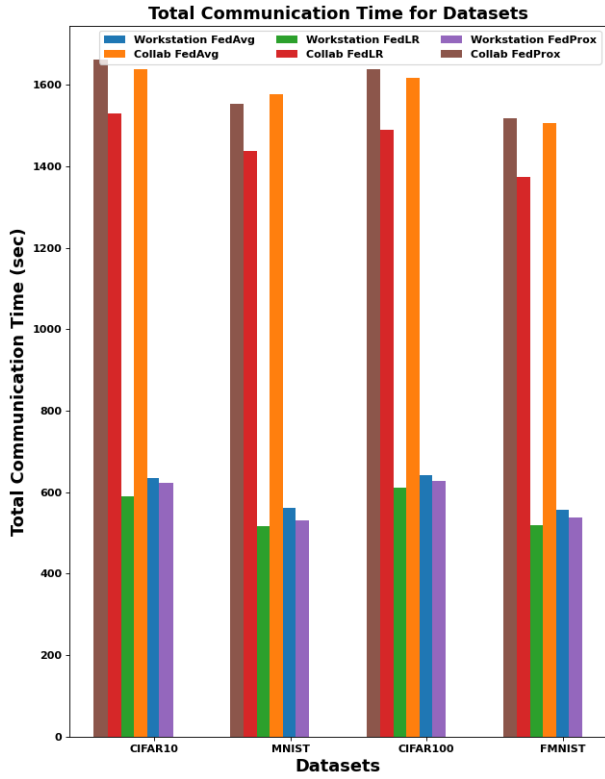
Fig. 12. Total Communication Time for Datasets.

Fig 4, 10, and 7 all show similar accuracy. This same pattern is seen in other datasets as well [25] [26] [22]. But fig 5, 8, and 11 shows the total individual times of each client, which is smallest in the case of FedLR. The detailed results can be found in table IV and the visual representation in fig. 12. In all the workstation results, FedLR has shown the best results, and FedProx is the second best. But in collab results, it was a mixed bag. As Collab gives you a random T4 GPU in each instance, which means the heterogeneity is more severe. Which lead to FedAvg performing better than FedProx in some cases. To interpret the results more easily Percentage

TABLE IV
TOTAL COMMUNICATION TIME WITH DIFFERENT DATASETS

Datasets	System	FedLR (sec)	FedAvg (sec)	FedProx (sec)
CIFAR10	Workstation	612.83	647.56	653.80
	Collab	1528.82	1638.73	1662.25
MNIST	Workstation	515.81	562.51	531.81
	Collab	1438.09	1576.74	1553.16
CIFAR100	Workstation	611.08	641.14	628.44
	Collab	1490.00	1617.62	1637.13
FMNIST	Workstation	519.84	557.56	537.23
	Collab	1374.31	1504.77	1516.53

difference has been calculated. Comparison between FedLR and FedAvg as well as FedLR and FedProx. Table V shows the detailed results. In the workstation setting, the proposed methodology has performed better than FedAvg and FedProx. However. as we have explained earlier, in the case of Google Collab, the results were a bit different. All tests have been run

several times, and the full results are available at Github [27].

TABLE V
PERCENTAGE DIFFERENCE OF COMMUNICATION TIME WITH DIFFERENT DATASETS

Datasets	Workstation FedAvg	Collab FedAvg	Workstation FedProx	Collab FedProx
CIFAR10	5.5%	6.9%	6.4%	8.3%
MNIST	8.7%	9.2%	3%	7.7%
CIFAR100	4.8%	8.2%	2.8%	9.4%
FMNIST	7.0%	9.0%	3.3%	9.8%

VI. CONCLUSION

The proposed algorithm has reduced the total communication time by a significant margin and has maintained fairness in the system while maintaining the accuracy of the models. The proposed algorithm has also been tested on two different devices to prove the improvements in communication rounds for a federated learning system with many slacker devices. The two devices prove the heterogeneity aspect of the proposed system. Both FedAvg and FedProx are compared with the proposed work, and the proposed work has significantly improved over both of them. Using the proposed methodology in real-world IoT and Edge devices will make the Federated Learning process more efficient and robust.

REFERENCES

[1] Yang Liu, Tao Fan, Tianjian Chen, Qian Xu, and Qiang Yang. Fate: An industrial grade platform for collaborative learning with data protection. *J. Mach. Learn. Res.*, 22(226):1–6, 2021.
[2] Nvidia. NVIDIA CLARA, 2020. https://www.nvidia.com/en-in/clara/ [Accessed 15-Dec-2024].
[3] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.
[4] Iot-Analytics. State of IoT 2024: Number of connected IoT devices growing 1318.8 billion globally, 2024. https://iot-analytics.com/number-connected-iot-devices/ [Accessed 15-Dec-2024].
[5] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
[6] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.
[7] Shuaijun Chen, Omid Tavallaie, Michael Henri Hambali, Seid Miad Zandavi, Hamed Haddadi, Song Guo, and Albert Y Zomaya. Boosting client selection of federated learning under device and data heterogeneity. *arXiv preprint arXiv:2310.08147*, 2023.
[8] Sakthi Vignesh Radhakrishnan, A Selcuk Uluagac, and Raheem Beyah. Gtid: A technique for physical device and device type fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 12(5):519–532, 2014.
[9] Kartik Patwari, Syed Mahbub Hafiz, Han Wang, Houman Homayoun, Zubair Shafiq, and Chen-Nee Chuah. Dnn model architecture fingerprinting attack on cpu-gpu edge devices. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 337–355. IEEE, 2022.
[10] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[11] Yingqi Zhang, Hui Xia, Shuo Xu, Xiangxiang Wang, and Lijuan Xu. Adaptfl: Adaptive federated learning framework for heterogeneous devices. *Future Generation Computer Systems*, page 107610, 2024.

[12] Weishan Zhang, Tao Zhou, Qinghua Lu, Yong Yuan, Amr Tolba, and Wael Said. Fedsl: A communication efficient federated learning with split layer aggregation. *IEEE Internet of Things Journal*, 2024.

[13] Zhenguo Ma, Yang Xu, Hongli Xu, Zeyu Meng, Liusheng Huang, and Yinxing Xue. Adaptive batch size for federated learning in resource-constrained edge computing. *IEEE Transactions on Mobile Computing*, 22(1):37–53, 2021.

[14] Fernanda C Orlandi, Julio CS Dos Anjos, Valderi RQ Leithardt, Juan Francisco De Paz Santana, and Claudio FR Geyer. Entropy to mitigate non-iid data problem on federated learning for the edge intelligence environment. *IEEE Access*, 11:78845–78857, 2023.

[15] Jed Mills, Jia Hu, and Geyong Min. Faster federated learning with decaying number of local sgd steps. *IEEE Transactions on Parallel and Distributed Systems*, 34(7):2198–2207, 2023.

[16] Akhil Mathur, Daniel J Beutel, Pedro Porto Buarque de Gusmão, Javier Fernandez-Marques, Taner Topal, Xinchi Qiu, Titouan Parcollet, Yan Gao, and Nicholas D Lane. On-device federated learning with flower. *arXiv preprint arXiv:2104.03042*, 2021.

[17] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[18] Vipul Singh Negi and Suchismita Chinara. Study of mobilenets model in federated learning. In *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 1040–1044. IEEE, 2023.

[19] Lingxi Hu, Yuanyuan Hu, Linhua Jiang, and Wei Long. Federated learning client selection algorithm based on gradient similarity. *Cluster Computing*, 28(2):134, 2025.

[20] Shengyuan Ye, Liekang Zeng, Qiong Wu, Ke Luo, Qingze Fang, and Xu Chen. Eco-fl: Adaptive federated learning with efficient edge collaborative pipeline training. In *Proceedings of the 51st International Conference on Parallel Processing*, pages 1–11, 2022.

[21] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[22] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[23] Markets and Markets. Edge AI Hardware Market, 2023. https://www.marketsandmarkets.com/Market-Reports/edge-ai-hardware-market-158498281.html [Accessed 22-Jul-2024].

[24] Yahya H Ezzeldin, Shen Yan, Chaoyang He, Emilio Ferrara, and A Salman Avestimehr. Fairfed: Enabling group fairness in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7494–7502, 2023.

[25] Yann LeCun. The mnist database of handwritten digits. *http://yann.lecun. com/exdb/mnist/*, 1998.

[26] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[27] Vipul Singh Negi. Github Code FedLR, 2024. https://github.com/vipulhld001/flower [Accessed 15-Dec-2024].