FEDSAW: Communication-Efficient Cross-Silo Federated Learning with Adaptive Compression

Chen Ying, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

Bo Li
Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Abstract—**Cross-silo federated learning (FL) is an emerging approach for institutions to collaboratively train a machine learning model without sharing their siloed data. However, it conventionally requires institutions to centrally store their clients' data, posing a threat to clients' privacy. This paper thus studies a preferable setting to leave data distributed to clients, where traditional FL is used not only across institutions but also among their clients. As this new setting inherits or even exacerbates the major problem of communication inefficiency in the traditional setting, we explore the feasibility of leveraging two compression techniques, pruning and quantization, to improve communication efficiency. Starting by applying off-the-shelf pruning and quantization mechanisms, we observe that they could largely reduce communication overhead with a negligible reduction, sometimes even a slight increase, in training performance. By mathematically analyzing the impact of compression on the performance of the trained model, we find that pruning and quantizing with a proper amount can offset possible performance degradation due to non-i.i.d data. Based on this finding, we propose FEDSAW, a new cross-silo FL framework that can improve communication efficiency by adaptively tuning the pruning amount and quantizing updates throughout training. In our extensive evaluation with six benchmark datasets, FEDSAW consistently outperformed its state-of-the-art competitors. It decreased the wall-clock training time and communication overhead used for converging to the target accuracy by up to 86.7% and 91.5%, respectively.**

I. INTRODUCTION

Since the performance of a machine learning model is strongly related to the amount of data available for its training, different *institutions* (*e.g.,* financial or medical organizations) share the incentive of leveraging their siloed data to train a global model with high performance [1]–[3]. However, under most circumstances, such data cannot be shared directly due to confidentiality or legal constraints. As a response, *cross-silo* federated learning (FL) [4] has been proposed as an appealing solution to enable institutions to collaboratively train a global model without sharing their privacy-sensitive siloed data.

It is routine for previous studies on cross-silo FL to consider a two-layer structure, only containing one central server and several institutions. They implicitly assume that each institution centrally stores raw data from a massive number — potential tens of thousands — of clients. Each client is likely an edge device, such as a mobile phone, generating large volumes of data. During one iteration of the global model training in the conventional cross-silo FL, each institution computes an updated model based on the current global model maintained by the central server with its siloed data, and only communicates this updated model with the central server.

The research was supported in part by a RGC RIF grant under the contract R6021-20, and RGC GRF grants under the contracts 16209120, 16200221 and 16207922.

Although data privacy within the confines of institutions can be well-preserved under such a two-layer structure, clients are under privacy risks. For clients in the financial, banking, and medical industries, their data should be kept private rather than stored within the same institution. Moreover, from the perspective of performance, sending raw data from clients to their institutions could take a much longer time than sending model updates. These above-mentioned issues motivate a three-layer structure, where traditional FL is used *both* across institutions *and* among their clients. Since institutions serve as edge servers in such a three-layer structure, we call them edge servers in the rest of this paper.

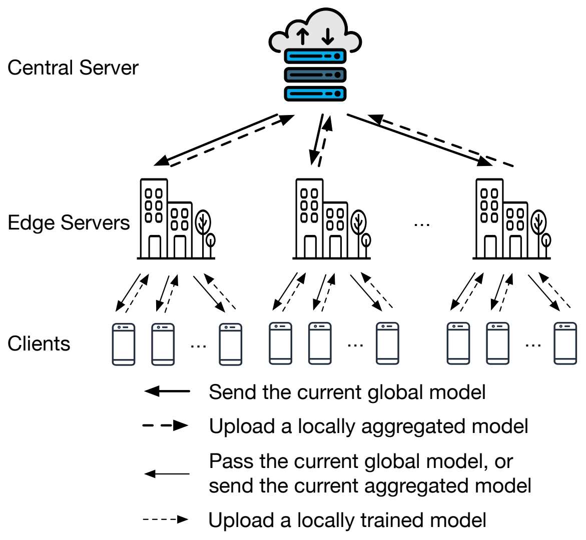

Central Server
Edge Servers
Clients
...
Send the current global model
Upload a locally aggregated model
Pass the current global model, or send the current aggregated model
Upload a locally trained model

Fig. 1: A three-layer structure of cross-silo FL.

In this paper, we focus on the new and more realistic context of cross-silo FL with a structure of three layers shown in Fig. 1. A global training iteration starts with the central server sending the current global model to edge servers, which pass it to their clients. These clients on the bottom layer then conduct local training to update the model with their local data and send their updated models to their edge servers. Each edge server on the middle layer aggregates its clients' models and sends its aggregated model back to its clients for several other rounds of local training on clients and local aggregation on itself. Finally, the edge server transfers its aggregated model to the central server. The central server then generates an improved global model by aggregating all edge servers' aggregated models.

However, a major defect, low communication efficiency, is buried in this three-layer structure. The exceedingly frequent exchanges of model weights throughout the training process may easily lead to massive communication overhead, due to the tremendous number of clients and large sizes of modern machine learning models. To make matters worse, limited and unstable network connections of clients, which are typical in FL, may further exacerbate the problem and result in a long training time. Additionally, it is common in the real world that clients' local datasets are non-i.i.d. (not independent and identically distributed). Existing works have proved that non-i.i.d. data could significantly reduce the accuracy and slow down the convergence of the global model [5], [6].

While abundant advanced methods have been proposed to improve the communication efficiency of FL, by and large, they only consider the traditional two-layer FL [7]–[9]. The three-layer cross-silo FL remains rarely explored and undoubtedly poses more challenges as it involves more communication. Moreover, we find that not all existing works on FL shared their implementations as open-source, and those open-source ones fell short of evaluating FL mechanisms appropriately. Hence, we start by implementing PLATO, an open-source research framework for scalable FL research from scratch, and use it to conduct extensive experiments of applying popular compression techniques in machine learning, pruning and quantization, to investigate if they can improve the communication efficiency in three-layer cross-silo FL.

Surprisingly, the results of our preliminary empirical study show that even an off-the-shelf pruning mechanism, L1-norm unstructured pruning, can significantly reduce communication overhead with negligible degradation in global model accuracy. With a simple adjustment, which is to prune local model updates instead of local model weights, the global model trained with pruning can converge to higher accuracy within a shorter amount of time. Quantization also indicates its potential of largely reducing the elapsed training time and communication overhead for the global model to reach a target accuracy. Given these observations, we mathematically analyze the impact of applying these compression techniques on the performance of the trained global model.

A notable remark of our analysis is that properly pruning and quantizing updates before sending them out for aggregation can effectively offset the accuracy reduction of the global model brought by non-i.i.d. data. We thus propose a new framework, FEDSAW, to improve communication efficiency and global model performance in three-layer cross-silo FL. As our analysis indicates that the weight difference between each edge server's aggregated model and the current global model is directly related to the global model performance, FEDSAW adaptively tunes each edge server's and its clients' pruning amount, i.e., the percentage of parameters in updates to be zeroed out, and decides whether to quantize pruned updates based on the weight difference between the edge server's aggregated model and the global model.

Original contributions. Highlights of our original contributions in this paper are as follows. *First*, as far as we know, we are the first to study communication efficiency in cross-silo FL *with three layers*. *Second*, we implement PLATO, an open-source research framework for scalable FL research from scratch, and conduct extensive experiments on it to motivate using pruning and quantization for higher communication efficiency. *Third*, as our empirical study indicates the promising potential of pruning and quantization on boosting communication efficiency with an insignificant reduction or even increase in global model accuracy, we further mathematically analyze the performance of the global model trained with these two compression techniques, and deduce that a proper pruning and quantization amount can offset the accuracy reduction brought by non-i.i.d. data. *Finally*, we design a new cross-silo FL framework, FEDSAW, to increase communication efficiency by adaptively tuning the pruning amount and quantizing updates of each edge server and its clients. Results of six benchmark datasets show that compared with its state-of-the-art competitor, FEDSAW can largely reduce the elapsed wall-clock training time and communication overhead for the global model to reach a target accuracy.

II. PRELIMINARIES

A. Three-Layer Cross-Silo FL

Suppose that there are N clients and M edge servers in total. Each client n, $n \in [N] = \{1, 2, \ldots, N\}$, stores a local dataset $\mathcal{D}_n$, which is a collection of training samples $\{\mathbf{x}_i, y_i\}_{i=1}^{|\mathcal{D}_n|}$. Every client belongs to one edge server. $\mathcal{C}^{(m)}$ denotes the set of edge server m's clients. For simplicity, we denote the union of the local datasets of edge server m's clients as $\mathcal{D}^{(m)} := \sqcup_{n\in\mathcal{C}^{(m)}}\mathcal{D}_n$, and the union of all local datasets as $\mathcal{D} := \sqcup_{m=1}^{M}\mathcal{D}^{(m)}$.

We consider a classification problem of S classes. The problem is defined over a compact space $\mathcal{X}$ and a label space $\mathcal{Y} = [S]$. Any data point $\{\mathbf{x}, y\}$ of a local dataset distributes over $\mathcal{X}\times\mathcal{Y}$. The goal of FL is to find the global model weights $\mathbf{w}$ that can characterize the output y_i given the input $\mathbf{x}_i$, $\{\mathbf{x}_i, y_i\} \in \mathcal{D}$, with the loss function $f_i(\mathbf{w})$. If $\mathcal{D}$ is centralized on the central server, the objective can be formulated as

$$\min_{\mathbf{w}} f(\mathbf{w}) \quad \text{where} \quad f(\mathbf{w}) := \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} f_i(\mathbf{w}). \tag{1}$$

However, the central server does not have access to $\mathcal{D}$. To generate the global model in three-layer cross-silo FL, we utilize the Federated Averaging (*FedAvg*) algorithm [10], the most frequently used algorithm in conventional two-layer FL, which combines local stochastic gradient descent (SGD) on each client with a central server that performs model averaging. Here, we still let each client conduct local SGD with its local epoch number of τ_c, minibatch size of B, and learning rate of η. But instead of directly communicating with the central server in *FedAvg*, the client sends its locally trained model to its edge server for local aggregation. In the t-th global training iteration and the t_e-th round of local aggregation at edge server m, its client n updates its local model $\mathbf{w}_n$ at its t_c-th local training epoch as:

$$\mathbf{w}_n(t, t_e, t_c) = \mathbf{w}_n(t, t_e, t_c - 1) - \eta \sum_{s=1}^{S} p_n(y = s) \cdot \nabla_{\mathbf{w}_n} \mathbb{E}_{\mathbf{x}|y=s} \big[\log f_s(\mathbf{x}, \mathbf{w}_n(t, t_e, t_c - 1)) \big], \quad (2)$$

where $t_c \in [\tau_c]$ and $t_e \in [\tau_e]$.

Each edge server m performs local aggregation after receiving locally trained models from its clients:

$$\mathbf{w}^{(m)}(t, t_e) = \sum_{n \in \mathcal{C}^{(m)}} \frac{|\mathcal{D}_n|}{|\mathcal{D}^{(m)}|} \mathbf{w}_n(t, t_e, \tau_c). \quad (3)$$

To generate a locally aggregated model that can contribute more to improve the global model accuracy, edge server m does not send its $\mathbf{w}^{(m)}(t, t_e)$ to the central server when $t_e < \tau_e$. Instead, it sends $\mathbf{w}^{(m)}(t, t_e)$ back to its clients, lets it be the initial model $\mathbf{w}_n(t, t_e + 1, 0)$ for their local SGD, and generates another aggregated model $\mathbf{w}^{(m)}(t, t_e + 1)$. An edge server will conduct this kind of local aggregation for τ_e rounds before sending $\mathbf{w}^{(m)}(t, \tau_e)$ to the central server.

After receiving aggregated updates from all the edge servers, the central server generates an improved global model:

$$\mathbf{w}(t) = \sum_{m=1}^{M} \frac{|\mathcal{D}^{(m)}|}{|\mathcal{D}|} \mathbf{w}^{(m)}(t, \tau_e). \quad (4)$$

If this new model satisfies a predefined condition such as finishing T training iterations or convergence, the training ends; otherwise, a new training iteration starts with the central server sending the current global model to all the edge servers.

B. *Pruning*

Pruning [11] is one of the most effective methods to compress machine learning models [12]. In general, neural networks are over-parameterized. Pruning can cut down the number of parameters to decrease model sizes without sacrificing their accuracy. Therefore, it has been considered a promising solution to improve communication efficiency in FL [13]–[17]. By pruning local models before transferring them, communication overhead can be sharply reduced.

Modern pruning techniques can be classified into structured (channel-level) and unstructured (parameter-level) pruning. We focus on unstructured pruning, which zeros out parameters that are less important to the model performance. There are different heuristics and methods to determine which parameters are less important and can be removed with minimal effect on model accuracy. One common method is to use magnitude, for example, L1 norm, to approximate the importance of each parameter. The intuition is that smaller magnitude parameters have smaller effects on the output, and hence are less likely to degrade model performance if they are pruned.

III. Motivation

This section presents our empirical study on applying pruning and quantization to three-layer cross-silo FL. The results motivated our new framework, FedSaw. The experimental implementation we used throughout this paper is within our open-source research framework Plato. Since a complete narrative on Plato is beyond the scope of this paper, we first present a brief introduction of it in this section, and then demonstrate our empirical study in detail.

A. Plato*: Our Open-Source Research Framework Created from Scratch*

As no existing works on cross-silo FL shared their implementations as open-source, and existing open-source FL frameworks fell short of evaluating cross-silo FL mechanisms appropriately, we have implemented Plato, an open-source research framework for scalable FL research from scratch, which is available at https://github.com/TL-System/plato. Development on Plato started in November 2020, and so far amounted to around 37 person-month of research and development time. Plato is designed and built with several key objectives: it is *scalable* to numerous clients, *extensible* to accommodate a wide variety of datasets, models, and FL algorithms; and *agnostic* to deep learning frameworks such as TensorFlow and PyTorch. In Plato, communication among the central server, edge servers, and clients is over industry-standard WebSockets. The central server may run on either the same GPU-enabled physical machine as its edge servers and clients, which is suitable for an emulation research testbed, or be deployed in a cloud datacenter.

B. *Cross-Silo FL with Pruning*

To explore the impact of pruning on the training performance of three-layer cross-silo FL with experiments, we study with four image classification tasks: the `LeNet-5` model with the `EMNIST` dataset and the `FEMNIST` (Federated Extended MNIST) dataset, the `ResNet-18` model with the `CIFAR-10` dataset, and the `VGG-16` model with the `CINIC-10` dataset.

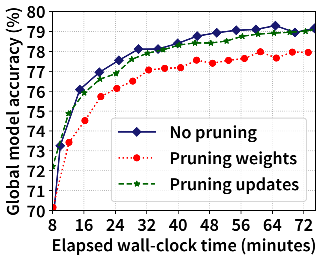
We conducted our experiments on NVIDIA A100 GPUs with 40 GB CUDA memory. Except for `FEMNIST` dataset which inherently provides 3597 local datasets with a highly skewed non-i.i.d. distribution, we sampled the other three datasets with the symmetric Dirichlet distribution with a concentration of 5 to generate non-i.i.d. local datasets. There are 5 edge servers, and each of them has the same number of clients. In every global training iteration, each edge server conducts 4 local aggregation rounds before sending its aggregated model to the central server. For local training of clients, we set the batch size to 32, epoch number to 5, learning rate to 0.01, and momentum to 0.9 in all our tasks. Other important parameters used across our tasks are listed in Table I, where $|\mathcal{C}^{(m)}|$ is the number of each edge server's selected clients.

TABLE I: Parameter settings.

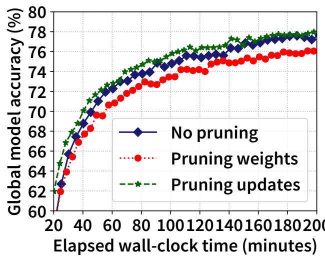
Parameter	`EMNIST`	`FEMNIST`	`CIFAR-10`	`CINIC-10`
N	1000	3597	1000	1000
$\lvert\mathcal{C}^{(m)}\rvert$	20	20	6	6
Weight decay	0	0	0.0001	0.0001

As a starting point, we use a simple pruning technique, L1-norm unstructured pruning, to zero out 40% model parameters with the lowest L1-norms for each client's locally trained model as well as each edge server's locally aggregated model. We also implement *Zstandard* [18], a fast real-time compression algorithm, in PLATO to compress data before communication, so that zeros in transferred models will take up almost no space. For example, 0.24 MB of data needs to be communicated when transferring a LeNet-5 model without pruning. After applying pruning and the *Zstandard* algorithm, clients and edge servers only need to send data of around 0.15 MB. Therefore, the communication overhead in each global training iteration is largely reduced, and so is the communication time of each global training iteration.

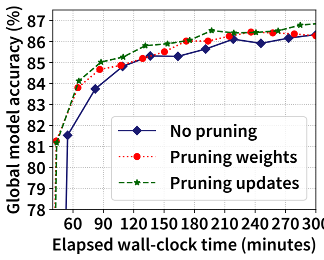
Although in *FedAvg* and most FL algorithms, client n sends $\mathbf{w}_n$, the weights of it locally trained model, we conjecture that when leveraging the L1-norm unstructured pruning, it would be more beneficial to prune and send updates of local models. That is, instead of pruning $\mathbf{w}_n(t, t_e, \tau_c)$ and sending the pruned local model to its edge server, client n sends pruned $\mathbf{w}_n(t, t_e, \tau_c) - \mathbf{w}^{(m)}(t, t_e - 1)$, the weight difference between its trained model and the model received from its edge server m before its local training. Likewise, rather than pruning and uploading $\mathbf{w}^{(m)}(t, t_e)$, edge server m sends pruned $\mathbf{w}^{(m)}(t, t_e) - \mathbf{w}(t - 1)$, the weight difference between its locally aggregated model and the model received from the central server at the beginning of this global training iteration. The intuition behind this conjecture is that training performance would decrease if the global model is aggregated with pruned weights of local models, which are different from the original weights before pruning, as a certain amount of them are zeroed out. While pruning updates only make some weights of local models revert to their values before local training or local aggregation, as if some weights of the global model have not been trained by some clients.



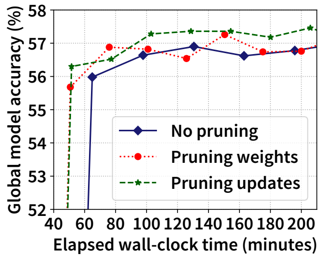
(a) `EMNIST` (`LeNet-5`)



(b) `FEMNIST` (`LeNet-5`)



(c) `CIFAR-10` (`ResNet-18`)



(d) `CINIC-10` (`VGG-16`)

Fig. 2: Performance comparison of no pruning, pruning model weights, and pruning model updates.

Fig. 2 illustrates the test accuracy of global model trained with or without pruning. The first observation is that the results strongly support our conjecture. Pruning model updates outperformed pruning model weights in all four tasks. It always reached higher test accuracy of the global model within the same amount of elapsed wall-clock training time.

Interestingly, these results also reveal that pruning model updates can sometimes increase global model accuracy. When training with the `FEMNIST`, `CIFAR-10`, and `CINIC-10` datasets, pruning model updates results in higher global model accuracy compared with no pruning. This finding may seem counterintuitive, as pruning zeros out some parameters in updates computed by local SGD, which is equivalent to not conducting the local training. However, the mathematical analysis presented in the next section well substantiates such accuracy improvements due to pruning model updates.

C. Cross-Silo FL with Quantization

In addition to pruning, there is another popular compression technique, quantization. Commonly, machine learning models use the format of 32-bit floating-point. If we quantize edge servers' aggregated updates and clients' updates to the 16-bit floating-point format, the communication overhead for transferring them can be cut by half.

However, as parameters are converted from 32-bit to 16-bit floating-point numbers, quantization may decrease the accuracy of the global model. Although the communication overhead in one round is reduced, it is possible that the global model needs more rounds to reach a target accuracy, which could increase the total communication overhead. Hence, we preliminarily test the feasibility of using quantization to improve communication efficiency in three-layer cross-silo FL.

We compare the performance of the basic cross-silo FL without pruning or quantization as the baseline, with L1-norm pruning, with quantization, and with L1-norm pruning and quantization in the same four tasks and experimental settings as Section III-B. Table II lists the elapsed time for the global model to reach a target accuracy. The numbers in brackets are the percentages of reduced elapsed training time of baseline.

Compared with baseline, quantization always reduced elapsed training time. When training with the `EMNIST` and `FEMNIST` datasets, applying both pruning and quantization had the best performance. In the task of training with the `CIFAR-10` dataset, just using quantization reduced 32% time of the baseline. However, using quantization in addition to pruning did not always result in the best performance. In the task of training with the `CIFAR-10` dataset, using pruning resulted in the best performance. While training with the `CINIC-10` dataset, although using pruning or quantization separately decreased training time, applying them at the same time failed to reach 57% accuracy, indicating possible degradation of combining these two compression methods in cross-silo FL.

Table III shows the performance of quantization regarding the communication overhead through a training session. Generally, quantization reduced communication overhead by a significant amount. For example, in the task of training the `VGG-16` model with the `CINIC-10` dataset, it reduced 78%

TABLE II: Elapsed wall-clock training time to reach a target accuracy with compression in different tasks.

Dataset	Model	Target Accuracy	Elapsed Training Time (Hours)			
			Baseline	Pruning	Quantization	Pruning + Quantization
EMNIST	LeNet-5	77%	0.42	0.47 (-13.16%)	0.29 (29.58%)	**0.24 (41.12%)**
FEMNIST	LeNet-5	73%	1.18	1.08 (8.32%)	0.83 (29.81%)	**0.69 (41.05%)**
CIFAR-10	ResNet-18	85%	2.27	**1.46 (35.70%)**	1.54 (32.06%)	1.78 (21.46%)
CINIC-10	VGG-16	57%	3.81	1.72 (54.97%)	**1.07 (71.98%)**	-

TABLE III: Communication overhead to reach a target accuracy with compression in different tasks.

Dataset	Communication Overhead (GB)			
	Baseline	Pruning	Quantization	Pruning + Quan
EMNIST	1.01	1.18	0.51 (49.42%)	**0.41 (59.58%)**
FEMNIST	2.90	2.73	1.46 (49.43%)	**1.18 (59.34%)**
CIFAR-10	53.37	35.75	**26.71 (49.96%)**	29.71 (44.33%)
CINIC-10	98.43	47.14	**21.10 (78.56%)**	-

communication overhead to reach 57% accuracy compared with training without quantization. However, it has the potential disadvantage of reducing global model accuracy.

Given that both pruning and quantization showed promising capabilities of improving the performance of cross-silo FL with three layers, we propose a new framework, FEDSAW to utilize these two compression techniques. However, such an improvement would be counterintuitive, as compression omits information in updates and may degrade the performance of the trained global model. Moreover, our experimental results show that using both pruning and quantization did not always lead to the best results. Therefore, we analyze the performance of the global model trained with pruning and quantization in the next section. Our mathematical analysis not only substantiates the seemly counterintuitive improvement due to compression, but also provides insights for designing FEDSAW to improve the communication efficiency.

IV. MATHEMATICAL ANALYSIS

One way to measure the global model performance is to compare the global model weights $\mathbf{w}(T)$ trained in FL with weights of a model $\mathbf{w}^*(T\tau_e\tau_c)$ trained by centralized SGD as if clients' data are centrally stored [5]. In such centralized SGD, the batch size is N times larger than that of local SGD, where N is the number of clients.

The centralized SGD updates its model in the t-th epoch as

$$\begin{aligned}\mathbf{w}^*(t) =&\mathbf{w}^*(t-1) - \eta\sum_{s=1}^{S} p(y=s)\cdot \\ &\nabla_{\mathbf{w}^*}\mathbb{E}_{\mathbf{x}|y=s}\big[\log f_s(\mathbf{x},\mathbf{w}^*(t-1))\big],\end{aligned} \tag{5}$$

where p is the probability for the data distribution of $\mathcal{D}$.

The ideal case for FL is that $\mathbf{w}(T)$ equals to $\mathbf{w}^*(T\tau_e\tau_c)$, where T is the total number of global training iterations. Here we define $\mathbf{w}^*(T,\tau_e,\tau_c) := \mathbf{w}^*(T\tau_e\tau_c)$ to be consistent with the format of local model $\mathbf{w}_n(T,\tau_e,\tau_c)$ of client n. Hence, the training performance of FL can be quantified by differences between $\mathbf{w}(T)$ and $\mathbf{w}^*(T,\tau_e,\tau_c)$, which we denote as $\Delta\mathbf{w}(T) := \|\mathbf{w}(T) - \mathbf{w}^*(T,\tau_e,\tau_c)\|$.

Assuming that $\nabla_{\mathbf{w}}\mathbb{E}_{\mathbf{x}|y=s}[\log f_s(\mathbf{x},\mathbf{w})]$ is $\lambda_{\mathbf{x}|y=s}$-Lipschitz for each class $s\in[S]$, we extend the analysis of traditional two-layer FL in [5] to bound $\Delta\mathbf{w}(T)$ in three-layer FL:

$$\begin{aligned}
&\Delta\mathbf{w}(T)\\
\overset{1}{\leq}&\sum_{m=1}^{M}\frac{|\mathcal{D}^{(m)}|}{|\mathcal{D}|}\sum_{n\in\mathcal{C}^{(m)}}\frac{|\mathcal{D}_n|}{|\mathcal{D}^{(m)}|}a_n\Delta\mathbf{w}_n(T,\tau_e,\tau_c-1)\\
\overset{2}{\leq}&\sum_{m=1}^{M}\frac{|\mathcal{D}^{(m)}|}{|\mathcal{D}|}\sum_{n\in\mathcal{C}^{(m)}}\frac{|\mathcal{D}_n|}{|\mathcal{D}^{(m)}|}\Big(a_n^{\tau_c}\Delta\mathbf{w}^{(m)}(T,\tau_e-1)\\
&+\eta\sum_{s=1}^{S}\|p_n(y=s)-p(y=s)\|\cdot\\
&\sum_{j=0}^{\tau_c-1}a_n^{j}g_{\max}(\mathbf{w}^*(T,\tau_e,\tau_c-1-j))\Big)\\
\overset{3}{\leq}&\sum_{m=1}^{M}\frac{|\mathcal{D}^{(m)}|}{|\mathcal{D}|}\sum_{n\in\mathcal{C}^{(m)}}\frac{|\mathcal{D}_n|}{|\mathcal{D}^{(m)}|}\Big(a_n^{\tau_c\tau_e}\Delta\mathbf{w}(T-1)\\
&+\eta\sum_{s=1}^{S}\|p_n(y=s)-p(y=s)\|\cdot\\
&\sum_{p=0}^{\tau_e-1}\sum_{j=0}^{\tau_c-1}a_n^{p\tau_c+j}g_{\max}(\mathbf{w}^*(T,\tau_e-p,\tau_c-1-j))\Big)\\
\overset{4}{\leq}&\sum_{m=1}^{M}\frac{|\mathcal{D}^{(m)}|}{|\mathcal{D}|}\sum_{n\in\mathcal{C}^{(m)}}\frac{|\mathcal{D}_n|}{|\mathcal{D}^{(m)}|}\Big(a_n^{T\tau_c\tau_e}\Delta\mathbf{w}(0)\\
&+\eta\sum_{s=1}^{S}\|p_n(y=s)-p(y=s)\|\cdot\\
&\sum_{q=0}^{T-1}\sum_{p=0}^{\tau_e-1}\sum_{j=0}^{\tau_c-1}a_n^{q\tau_e\tau_c+p\tau_c+j}.\\
&g_{\max}(\mathbf{w}^*(T-q,\tau_e-p,\tau_c-1-j))\Big),
\end{aligned} \tag{6}$$

where $\Delta\mathbf{w}_n(t,t_e,t_c) := \|\mathbf{w}_n(t,t_e,t_c) - \mathbf{w}^*(t,t_e,t_c)\|$, $\Delta\mathbf{w}^{(m)}(t,t_e) := \|\mathbf{w}^{(m)}(t,t_e) - \mathbf{w}^*(t,t_e,\tau_c)\|$, $a_n = 1+\eta\sum_{s=1}^{S}p_n(y=s)\lambda_{x|y=s}$ and $g_{\max}(\mathbf{w}) = \max_{s=1}^{S}\|\nabla_{\mathbf{w}}\mathbb{E}_{x|y=s}\log f_s(x,\mathbf{w})\|$.

The fourth inequality in Eq. (6) indicates that even the initial weights of $\mathbf{w}$ and $\mathbf{w}^*$ are the same, their differences will accumulate through training due to $\sum_{s=1}^{S}\|p_n(y=s)-p(y=s)\|$, the probability distance from the data distribution of each client to the data distribution of $\mathcal{D}$.

The third inequality shows that two parts amplify $\Delta\mathbf{w}(T)$. One is $\Delta\mathbf{w}(T-1)$, the weight difference in the last global training iteration. The other is $\sum_{s=1}^{S} \|p_n(y=s)-p(y=s)\|$. However, as $\sum_{s=1}^{S} \|p_n(y=s)-p(y=s)\|$ is fixed and unknown, we cannot use it to minimize $\Delta\mathbf{w}(T)$. Given that $\Delta\mathbf{w}(T-1)$ also affects $\Delta\mathbf{w}(T)$, one potential direction is to minimize $\Delta\mathbf{w}(t)$ in each iteration t.

To minimize $\Delta\mathbf{w}(t)$, the first and the second inequalities give some hints. If $\Delta\mathbf{w}_n(t,t_e,t_c)$ and $\Delta\mathbf{w}^{(m)}(t,t_e)$ are minimized, $\Delta\mathbf{w}(t)$ can be minimized as well. This explains the observations in our previous empirical study. The performance degradation when pruning model weights is because that some parameters of $\mathbf{w}_n(t,t_e,t_c)$ and $\mathbf{w}^{(m)}(t,t_e)$ are changed to zero, which increases $\Delta\mathbf{w}_n(t,t_e,t_c)$ and $\Delta\mathbf{w}^{(m)}(t,t_e)$ and thus results in larger $\Delta\mathbf{w}(t)$ and $\Delta\mathbf{w}(T)$. It also explains the interesting while counterintuitive observation that pruning and quantizing model updates could improve global model performance given non-i.i.d. data.

Suppose that client n prunes its local model updates with its pruning mask $\psi_n \in \{0,1\}^{|\mathbf{w}_n|}$, and thus its model weights aggregated by its edge server m is $\tilde{\mathbf{w}}_n(t,t_e,\tau_c) := \psi_n \odot (\mathbf{w}_n(t,t_e,\tau_c) - \mathbf{w}^{(m)}(t,t_e-1)) + \mathbf{w}^{(m)}(t,t_e-1)$. Edge server m also has its pruning mask $\psi^{(m)} \in \{0,1\}^{|\mathbf{w}^{(m)}|}$. Its locally aggregated model that will be aggregated by the central server becomes $\tilde{\mathbf{w}}^{(m)}(t,\tau_e) := \psi^{(m)} \odot (\mathbf{w}^{(m)}(t,\tau_e) - \mathbf{w}(t-1)) + \mathbf{w}(t-1)$. If in each global training iteration t, $\|\tilde{\mathbf{w}}_n(t,t_e,\tau_c) - \mathbf{w}^*(t,t_e,\tau_c)\| < \Delta\mathbf{w}_n(t,t_e,\tau_c)$ and $\|\tilde{\mathbf{w}}^{(m)}(t,\tau_e) - \mathbf{w}^*(t,\tau_e,\tau_c)\| < \Delta\mathbf{w}^{(m)}(t,\tau_e)$, $\Delta\mathbf{w}(t)$ will be decreased, and it will improve the trained global model $\mathbf{w}(T)$. Hence, by properly adjusting masks $\psi^{(m)}$ and ψ_n, $\forall m \in [M], \forall n \in \mathcal{C}^{(m)}$, through the training process, pruning can increase global model accuracy along with reducing communication overhead. Similarly, if quantization can further decrease $\|\tilde{\mathbf{w}}_n(t,t_e,\tau_c) - \mathbf{w}^*(t,t_e,\tau_c)\|$ and $\|\tilde{\mathbf{w}}^{(m)}(t,\tau_e) - \mathbf{w}^*(t,\tau_e,\tau_c)\|$, the training performance can be improved even more. This insight motivates us to propose FedSaw, a new framework for cross-silo FL with three layers.

V. FedSaw: Design and Workflow

This section presents our proposed framework, FedSaw, in detail. As our empirical and theoretical studies illustrate the promising potential of utilizing pruning and quantization to reduce communication overhead with possible improvement on model performance, we first present the design of FedSaw that can improve both the communication efficiency and global model accuracy, and then show its workflow.

A. Design

Our analysis in the last section reveals that by minimizing $\|\tilde{\mathbf{w}}^{(m)}(t,\tau_e) - \mathbf{w}^*(t,\tau_e,\tau_c)\|$ in each global training iteration t and $\|\tilde{\mathbf{w}}_n(t,t_e,\tau_c) - \mathbf{w}^*(t,t_e,\tau_c)\|$ in each local aggregation round t_e, pruning can offset the accuracy reduction on the global model due to training with non-i.i.d. data. However, as clients' data are not centrally stored for training $\mathbf{w}^*$, $\mathbf{w}^*(t,t_e,t_c)$ is unknown. A feasible workaround is to use $\mathbf{w}(t)$ and $\mathbf{w}^{(m)}(t,t_e)$ to substitute $\mathbf{w}^*(t,t_e,t_c)$, and accordingly change the objective to minimizing $\|\tilde{\mathbf{w}}^{(m)}(t,\tau_e) - \mathbf{w}(t)\|$ and $\|\tilde{\mathbf{w}}_n(t,t_e,\tau_c) - \mathbf{w}^{(m)}(t,t_e)\|$. The intention is that decreasing the weight difference between every local model and the model aggregated with it could potentially decrease this weight difference after the next aggregation.

Hence, for edge server m, if $\|\tilde{\mathbf{w}}^{(m)}(t,\tau_e) - \mathbf{w}(t)\|$ is large, its pruning mask $\psi^{(m)}(t+1)$ in the next global training iteration should contain more zeros, i.e., with a higher pruning amount $\rho^{(m)}(t+1)$, so that $\|\tilde{\mathbf{w}}^{(m)}(t+1,\tau_e) - \mathbf{w}(t+1)\|$ would likely decrease and in turn leads to smaller $\Delta\mathbf{w}(T)$.

Similarly, it is intuitive to adjust the pruning amount ρ_n of client n to minimize $\|\tilde{\mathbf{w}}_n(t,t_e,\tau_c) - \mathbf{w}^{(m)}(t,t_e-1)\|$ in each round of its edge server's local aggregation as well. And the larger the weight difference $\|\tilde{\mathbf{w}}_n(t,t_e,\tau_c) - \mathbf{w}^{(m)}(t,t_e-1)\|$, the higher the pruning amount ρ_n. However, it is common in the real world that there are a massive number of clients. And due to unstable and limited network communication, it is very likely that a client is able to participate at most once. That is, even if we compute ρ_n to generate ψ_n for client n who participated in the last round of aggregation, it may not conduct local training again. Thus, we use $\rho^{(m)}$ as the pruning amount of all the clients of edge server m.

Therefore, we design FedSaw to adaptively tune the pruning amount of each edge server and its clients with the rule that edge server m with larger $\|\tilde{\mathbf{w}}^{(m)}(t,\tau_e) - \mathbf{w}(t)\|$ should have higher pruning amount.

To further decrease $\|\tilde{\mathbf{w}}_n(t,t_e,\tau_c) - \mathbf{w}^{(m)}(t,t_e-1)\|$ and $\|\tilde{\mathbf{w}}^{(m)}(t,\tau_e) - \mathbf{w}(t)\|$, we quantize updates of edge servers and their clients with large $\|\tilde{\mathbf{w}}^{(m)}(t,\tau_e) - \mathbf{w}(t)\|$ after pruning these updates and before sending them out.

With extensive empirical study, we find that at the end of the t-th global training iteration, setting $\rho^{(m)}(t+1)$, the pruning amount of edge server m and its clients in the next global training iteration, as below could improve the training performance:

$$\rho^{(m)}(t+1) = \text{sigmoid}\left(\frac{\Delta\tilde{\mathbf{w}}^{(m)}(t,\tau_e) - \varpi}{\varpi}\right), \qquad (7)$$

where $\Delta\tilde{\mathbf{w}}^{(m)}(t,\tau_e) := \|\tilde{\mathbf{w}}^{(m)}(t,\tau_e) - \mathbf{w}(t)\|$ and ϖ is the median of $\{\Delta\tilde{\mathbf{w}}^{(m)}(t,\tau_e)\}$. In addition, quantizing pruned updates of edge server m and its clients in the next global training iteration if $\Delta\tilde{\mathbf{w}}^{(m)}(t,\tau_e) > \varpi$ can enhance the training performance.

To generate pruning mask $\psi^{(m)}$ and ψ_n, $\forall n \in \mathcal{C}^{(m)}$ with the pruning amount $\rho^{(m)}$ computed by Eq. (7), one option is to use L1-norm unstructured pruning, which shows its superiority in Section III-B. Also, a recent work [19] indicates that random pruning can speed up the sparse training of modern neural networks. As it is hard to theoretically analyze which parameters are truly important to the model performance so that they should not be zeroed out, randomly zeroing out parameters might be better than doing so with certain pruning criteria. To explore if random pruning is also powerful in three-layer cross-silo FL, we implement it in FedSaw and conduct an empirical study to compare it with L1-norm pruning.

B. Workflow

Below is the workflow of FedSaw. Since every edge server follows the same workflow in parallel, here we only demonstrate the steps conducted by edge server m, its clients, and the central server in the t-th global training iteration. Repeat the following Step 1 to Step 6 until the global model converges or reaches a target accuracy.

Step 1: The central server sends the current global model $\mathbf{w}(t-1)$, pruning amount $\rho^{(m)}(t)$, and whether quantization should be conducted to edge server m. In the first global training iteration, $\mathbf{w}(0)$ is randomly generated and $\rho^{(m)}(1) = \rho$.

Step 2: Edge server m uses this model $\mathbf{w}(t-1)$ as its initial aggregated model $\mathbf{w}^{(m)}(t, 0)$ at the beginning of its first round of local aggregation. For the other t_e-th round, edge server m sends its locally aggregated model $\mathbf{w}^{(m)}(t, t_e-1)$ and pruning amount $\rho^{(m)}(t)$ to its clients, and also notifies them if they should quantize their pruned updates.

Step 3: After receiving $\mathbf{w}^{(m)}(t, t_e-1)$ from its edge server, client n, $\forall n \in \mathcal{C}^{(m)}$, conducts local SGD in parallel to generate its local model $\mathbf{w}_n(t, t_e, \tau_c)$. After pruning updates $\mathbf{w}_n(t, t_e, \tau_c) - \mathbf{w}^{(m)}(t, t_e - 1)$ with the pruning amount of $\rho^{(m)}(t)$ to compute $\tilde{\mathbf{w}}_n(t, t_e, \tau_c)$ and quantizing it if required, client n sends it to its edge server m.

Step 4: Edge server m decompresses $\tilde{\mathbf{w}}_n(t, t_e, \tau_c)$ upon receiving it. After receiving pruned updates from all its clients, edge server m aggregates them to generate its new local model $\mathbf{w}^{(m)}(t, t_e)$. Steps 2 to 4 are repeated until edge server m finishes τ_e rounds of local aggregation.

Step 5: After finishing its τ_e-th round of local aggregation, edge server m prunes $\mathbf{w}^{(m)}(t, \tau_e) - \mathbf{w}(t-1)$ with the pruning amount of $\rho^{(m)}(t)$ to generate $\tilde{\mathbf{w}}^{(m)}(t, \tau_e)$, which will be quantized if required, and sent to the central server.

Step 6: Once receiving compressed aggregated models from all the edge servers, the central server decompresses and aggregates them to generate $\mathbf{w}(t)$, computes $\rho^{(m)}(t+1)$ with Eq. (7), and determines whether quantization should be conducted in the next round for each edge server m.

VI. Evaluation

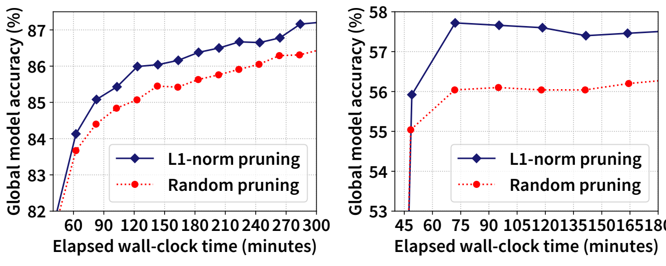
We have implemented FedSaw in Plato and evaluated it with six widely used datasets and three popular models in the machine learning field, the `MNIST`, `FashionMNIST`, `EMNIST`, and `FEMNIST` datasets with the `LeNet-5` model, the `CIFAR-10` dataset with the `ResNet-18` model, and the `CINIC-10` dataset with the `VGG-16` model.

Experimental settings are the same as that in our previous experimental study presented in Section III. For tasks of training the `LeNet-5` model with `MNIST` and `FashionMNIST` datasets, the parameter settings are the same as training the `LeNet-5` model with `EMNIST` dataset shown in Table I.

We first investigate whether the L1-norm pruning or random pruning is better for FedSaw to improve communication efficiency in cross-silo FL. After deciding the suitable strategy to generate the pruning mask, we apply it and compare FedSaw with *Sub-FedAvg* [14], a state-of-the-art framework for improving the communication efficiency of FL. It iteratively prunes the parameters and channels of each client's local model during its local training. Since *Sub-FedAvg* was designed for two-layer FL, for a fair comparison, we let edge servers prune their aggregated models after each local aggregation by following the same pruning strategy used by clients in our experiments. We use 0.4 as the initial pruning amount of *Sub-FedAvg* and FedSaw.

A. FedSaw*: L1-Norm Pruning or Random Pruning*

As random pruning is also a popular pruning strategy and a recent work [19] claimed its practicability in machine learning, we testify if it is a better fit to FedSaw than the L1-norm pruning, which shows its effectiveness in our previous empirical study in Section III-B.



(a) `CIFAR-10` (`ResNet-18`)   (b) `CINIC-10` (`VGG-16`)

Fig. 3: Comparing the performance of FedSaw with L1-norm pruning and with random pruning.

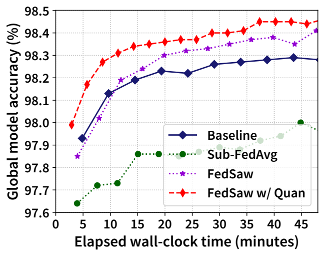
Unfortunately, the performance of FedSaw with random pruning is not satisfactory. In all our experiments, L1-norm pruning always outperformed random pruning. Fig. 3 shows the global model accuracy as training time elapsed through training with `CIFAR-10` and `CINIC-10` datasets as examples.

These results can be explained with the design rule of FedSaw. Higher pruning amounts are applied to larger local updates to make their magnitudes closer to 0, as our analysis in Section IV indicates that these local updates would degrade global model performance. However, there are chances that some parameters in these local updates could actually improve the global model performance. Hence, to maintain the possible contribution of these updates to the global model, zeroing out parameters with smaller L1-norm values but keeping those with larger L1-norm values unchanged should be better than randomly selecting parameters to zero out. Therefore, L1-norm pruning is a better choice for FedSaw than random pruning.

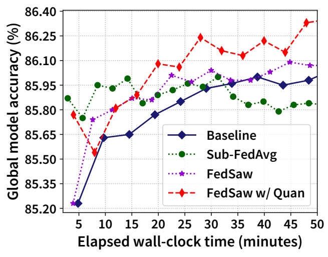
B. *Comparing* FedSaw *with the State-of-the-Art*

To further evaluate the performance of FedSaw with L1-norm pruning and quantization, we compare it with *Sub-FedAvg* and no pruning and no quantization as the baseline. Fig. 4 demonstrates global training accuracy with the elapsed wall-clock time through training in the six tasks. To better evaluate pruning and quantization separately, the dotted purple lines with star markers are for FedSaw with adaptive pruning only, while the dashed red lines with thin diamond markers are for FedSaw with both adaptive pruning and quantization.

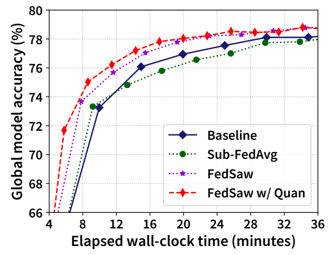
Generally, even only with adaptive pruning, FedSaw required less elapsed training time than the baseline and *Sub-FedAvg* to reach the same accuracy. With quantization, FedSaw saved even more time.



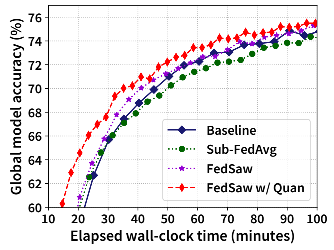
(a) MNIST (LeNet-5)



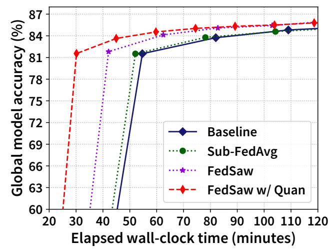
(b) FashionMNIST (LeNet-5)



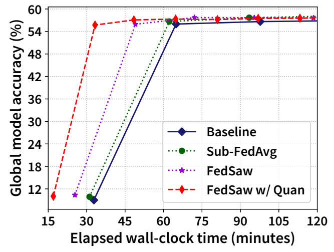
(c) EMNIST (LeNet-5)



(d) FEMNIST (LeNet-5)



(e) CIFAR-10 (ResNet-18)



(f) CINIC-10 (VGG-16)

Fig. 4: Performance comparison: FEDSAW vs. its competitors.

Another observation is that the performance of *Sub-FedAvg* is not satisfactory for three-layer cross-silo FL. In the tasks of training with the CIFAR-10 and CINIC-10 datasets, *Sub-FedAvg* only slightly outperformed the baseline. While in the other four tasks, its performance was worse than the baseline.

A possible explanation of the unsatisfactory performance of *Sub-FedAvg* is that it was designed for traditional two-layer FL. Given that we already made the necessary adjustment to use it under our three-layer structure, its performance degradation shows the difference between two-layer and three-layer FL and the difficulty to improve performance under this practical while rarely studied three-layer structure.

Table IV and Table V list all the numerical results regarding elapsed wall-clock training time and communication overhead, respectively. Numbers in bracelets are the percentages of reduction in elapsed wall-clock training time and communication overhead for the global model to reach the target accuracy compared with the baseline. In three-layer cross-silo FL, *Sub-FedAvg* failed to have better performance than baseline in half of the six evaluated tasks. While FEDSAW consistently improved the performance of baseline even with adaptive pruning only. With quantization, FEDSAW always had the best performance among the four evaluated mechanisms. Especially in the task of training the VGG-16 model with the CINIC-10 dataset, FEDSAW reduced 78.82% training time and 83.42% communication overhead compared to the baseline. When training the LeNet-5 model with the MNIST dataset, FEDSAW with quantization reduced 86.67% training time and 91.53% communication overhead compared to *Sub-FedAvg*.

VII. Related Work

Since communication is a key bottleneck in FL, various communication-efficient methods have been proposed. A common direction is to use compression techniques to reduce the size of messages communicated in each training iteration [20]. As two of the most common techniques in the literature to reduce the computational and memory requirements of neural networks, pruning and quantization have been used in several works to improve communication efficiency in FL.

With the assumption that the central server knows the channel state information of all clients, Liu et al., [15] formulate an optimization problem to compute the optimal pruning amount to maximize the convergence rate under a given learning rate budget. In their next work [17], the optimization problem solves the optimal client selection in addition to the pruning amount. However, their assumption is not always practical. The channel state information of each client is hard to know beforehand and could vary from time to time during training.

To minimize the total training time of FL, *PruneFL* [13] first selects one trusted client with high computational capabilities to start training with a small model. During training, both the server and clients conduct adaptive pruning. *PruneFL* formulates the problem of finding optimal pruning masks as minimizing an approximated risk reduction divided by an approximated time of one training iteration. A problem with this approach is that it is hard to find an optimal client and approximate the time of each iteration in reality, considering numerous clients and unstable network connections.

Another FL framework *FL-PQSU* [16] combines pruning, weight quantization, and letting clients only upload their updates when their training loss in this global training iteration is higher than the previous iteration. The pruning method it uses is the L1-norm structured pruning with a predefined pruning amount. All the above-mentioned existing works are for the traditional two-layer FL, while the three-layer cross-silo FL we study throughout this paper remains uncharted territory.

VIII. Concluding Remarks

In this paper, we focus on a rarely studied while practical three-layer structure of cross-silo FL, where training data are kept on clients' sides rather than centrally stored in institutions. We first implement PLATO, our open-source FL framework, and explore the effect of using pruning in three-layer cross-silo FL through empirical study. With the observation that pruning and quantization can largely improve communication efficiency with negligibly reducing or even increasing global model accuracy, we mathematically analyze the training performance with these two compression techniques. As our theoretical study indicates that pruning and quantization can improve the global model performance if compressing updates with the right amount to offset the weight differences due to non-i.i.d. local data, we propose a new framework, FEDSAW,

TABLE IV: Elapsed wall-clock training time to reach a target accuracy in different tasks.

Dataset	Model	Target Accuracy	Elapsed Training Time (Hours)			
			Baseline	*Sub-FedAvg*	FEDSAW	FEDSAW + Quantization
`MNIST`	`LeNet-5`	98%	0.16	0.75 (-363.44%)	0.13 (17.83%)	**0.10 (40.87%)**
`FashionMNIST`	`LeNet-5`	86%	0.65	0.52 (19.23%)	0.38 (41.87%)	**0.33 (48.47%)**
`EMNIST`	`LeNet-5`	77%	0.42	0.43 (-2.92%)	0.26 (37.97%)	**0.24 (42.62%)**
`FEMNIST`	`LeNet-5`	75%	1.76	2.02 (-14.56%)	1.65 (6.64%)	**1.46 (17.06%)**
`CIFAR-10`	`ResNet-18`	85%	2.27	2.16 (4.91%)	1.38 (39.30%)	**1.24 (45.39%)**
`CINIC-10`	`VGG-16`	57%	3.81	1.56 (59.13%)	1.20 (68.48%)	**0.81 (78.82%)**

TABLE V: Communication overhead to reach a target accuracy in different tasks.

Dataset	Model	Target Accuracy	Communication Overhead (GB)			
			Baseline	*Sub-FedAvg*	FEDSAW	FEDSAW + Quantization
`MNIST`	`LeNet-5`	98%	0.39	1.77 (-357.13%)	0.32 (16.25%)	**0.15 (60.13%)**
`FashionMNIST`	`LeNet-5`	86%	1.55	0.83 (46.03%)	0.91 (40.88%)	**0.81 (47.55%)**
`EMNIST`	`LeNet-5`	77%	1.01	1.05 (-3.69%)	0.65 (35.84%)	**0.40 (60.90%)**
`FEMNIST`	`LeNet-5`	75%	4.34	4.79 (-10.26%)	4.00 (7.93%)	**2.33 (46.33%)**
`CIFAR-10`	`ResNet-18`	85%	53.37	51.31 (3.87%)	34.09 (36.12%)	**20.58 (61.44%)**
`CINIC-10`	`VGG-16`	57%	98.43	40.55 (58.81%)	33.71 (65.75%)	**16.32 (83.42%)**

to improve both communication efficiency and global model performance in cross-silo FL with three layers by adaptively tuning the pruning amount of each institution and its clients and determining whether to quantize their pruned updates. Our experimental results demonstrate the effectiveness of FEDSAW under various training tasks. It consistently and significantly outperformed its competitors in terms of reducing elapsed wall-clock training time and communication overhead.

REFERENCES

[1] J. Ma, Q. Zhang, J. Lou, J. C. Ho, L. Xiong, and X. Jiang, "Privacy-Preserving Tensor Factorization for Collaborative Health Data Analysis," in *Proc. 28th ACM International Conference on Information and Knowledge Management (CIKM)*, 2019, pp. 1291–1300.

[2] D. Gao, C. Ju, X. Wei, Y. Liu, T. Chen, and Q. Yang, "HHHFL: Hierarchical Heterogeneous Horizontal Federated Learning for Electroencephalography," in *Proc. Int'l Workshop on Federated Machine Learning for User Privacy and Data Confidentiality, in Conjunction with IJCAI 2019 (FL-IJCAI'2019)*, 2019.

[3] P. Courtiol, C. Maussion, M. Moarii, E. Pronier, S. Pilcer, M. Sefta, P. Manceron, S. Toldo, M. Zaslavskiy, N. Le Stang, N. Girard, O. Elemento, A. G. Nicholson, J.-Y. Blay, F. Galateau-Sallé, G. Wainrib, and T. Clozel, "Deep Learning-Based Classification of Mesothelioma Improves Prediction of Patient Outcome," *Nature Medicine*, vol. 25, no. 10, pp. 1519–1525, October 2019.

[4] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and Open Problems in Federated Learning," *arXiv preprint arXiv:1912.04977*, 2019.

[5] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated Learning with Non-IID Data," *arXiv preprint arXiv:1806.00582*, 2018.

[6] F. Sattler, S. Wiedemann, K.-R. Muller, and W. Samek, "Robust and Communication-Efficient Federated Learning from Non-IID Data," *IEEE Trans. Neural Networks and Learning Systems*, 2019.

[7] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," in *Proc. Neural Information Processing Systems (NIPS) Workshop on Private Multi-Party Machine Learning (PMPML)*, 2016.

[8] J. Sun, T. Chen, G. B. Giannakis, and Z. Yang, "Communication-Efficient Distributed Learning via Lazily Aggregated Quantized Gradients," in *Neural Information Processing Systems (NeurIPS)*, 2019.

[9] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated Learning with Matched Averaging," in *Proc. 8th International Conference on Learning Representations (ICLR)*, 2020.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 54, April 2017, pp. 1273–1282.

[11] M. C. Mozer and P. Smolensky, "Using Relevance to Reduce Network Size Automatically," *Connection Science*, vol. 1, no. 1, pp. 3–16, 1989.

[12] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both Weights and Connections for Efficient Neural Network," in *Proc. 29th International Conference on Neural Information Processing Systems (NeurIPS)*, 2015, pp. 1135–1143.

[13] Y. Jiang, S. Wang, B. J. Ko, W. Lee, and L. Tassiulas, "Model Pruning Enables Efficient Federated Learning on Edge Devices," in *Proc. NeurIPS-20 Workshop on Scalability, Privacy, and Security in Federated Learning (SpicyFL)*, 2010.

[14] S. Vahidian, M. Morafah, and B. Lin, "Personalized Federated Learning by Structured and Unstructured Pruning under Data Heterogeneity," in *41st IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 2021, pp. 27–34.

[15] S. Liu, G. Yu, R. Yin, and J. Yuan, "Adaptive Network Pruning for Wireless Federated Learning," *IEEE Wirel. Commun. Lett.*, vol. 10, no. 7, pp. 1572–1576, 2021.

[16] W. Xu, W. Fang, Y. Ding, M. Zou, and N. Xiong, "Accelerating Federated Learning for IoT in Big Data Analytics With Pruning, Quantization and Selective Updating," *IEEE Access*, vol. 9, pp. 38 457–38 466, 2021.

[17] S. Liu, G. Yu, R. Yin, J. Yuan, L. Shen, and C. Liu, "Joint Model Pruning and Device Selection for Communication-Efficient Federated Edge Learning," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 231–244, 2022.

[18] Y. Collet, N. Terrell, and P. Skibiński, "Zstandard," 2021. [Online]. Available: https://github.com/facebook/zstd

[19] S. Liu, T. Chen, X. Chen, L. Shen, D. C. Mocanu, Z. Wang, and M. Pechenizkiy, "The Unreasonable Effectiveness of Random Pruning: Return of the Most Naive Baseline for Sparse Training," in *Proc. 10th International Conference on Learning Representations (ICLR)*, 2022.

[20] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020.