An Efficient and Accurate Dynamic Sparse Training Framework Based on Parameter-Freezing

Lei Li[1*], Haochen Yang[1*], Jiacheng Guo[1], Hongkai Yu[1], Minghai Qin[1,2†], Tianyun Zhang[1†]

[1]Cleveland State University, Cleveland, USA
[2]Western Digital Research, Milpitas, USA

Abstract

Federated learning is a decentralized machine learning approach that consists of servers and clients. It protects data privacy during model training by keeping the training data locally in each client. However, the requirement for the server and clients to frequently synchronize the parameters of the model brings a heavy burden to the communication links, especially when the model size has grown drastically in recent years. Several methods have been proposed to compress the model size by sparsification to reduce the communication overhead, albeit with significant accuracy degradation. In this work, we propose methods to better trade-off between model accuracy and training efficiency in federated learning. Our first proposed method is a novel sparse mask readjustment rule on the server and the second is a parameter-freezing method during training on the clients. Experimental results show that the model accuracy has significantly improved when combining our proposed methods. For example, compared with the previous state-of-the-art methods with the same total amount of communication cost and computation FLOPs, the accuracy increases on average by 4% and 6% in our methods for CIFAR-10 and CIFAR-100 datasets on ResNet-18, respectively. On the other hand, when targeting the same accuracy, the proposed method can reduce the communication cost by 4-8 times for different datasets with different sparsity levels.

Introduction

In the era of ubiquitous computing, training deep neural networks (DNNs) often requires a substantial amount of data, typically generated at the edge of widely used mobile and Internet of Things (IoT) devices (Zhang, Patras, and Haddadi 2019; Mohammadi et al. 2018; Li, Ota, and Dong 2018). This raises significant privacy concerns as centralizing such data for training purposes poses risks to user confidentiality (Voigt and Von dem Bussche 2017; Khan et al. 2021; Nguyen et al. 2021). To deal with these challenges, Federated Learning (FL) has emerged as a pivotal distributed machine learning paradigm. Developed by (McMahan et al. 2017) and further explored by (Kairouz et al. 2021), FL enables multiple clients to collaboratively train a global model without sharing local data. Instead, these clients contribute by aggregating locally trained parameters, thereby preserving privacy and harnessing the collective power of data distributed across numerous devices.

In FL systems, substantial computational tasks are transferred from centralized cloud servers to edge devices with constrained resources. To facilitate effective operation at the edge, an FL system needs to enhance both the efficiency of local training on the devices and the efficiency of data communication across the network. Current DNNs have large model sizes and computational requirements which pose challenges for deployment and training on edge devices. Some research has explored the use of sparse neural networks to create lightweight models for edge devices, aiming to reduce inference latency(Gale, Elsen, and Hooker 2019; Chen et al. 2020, 2021c). On the other hand, implementing FL frameworks on client devices such as mobile phones often faces severe upload bandwidth limitations, making it equally crucial to reduce the upload costs of FL algorithms. Prior work mitigates these constraints by condensing the models or messages into compact formats, such as gradient compression (Lin et al. 2018; Vogels, Karimireddy, and Jaggi 2019; Albasyoni et al. 2020). Similarly, parameter-freezing offers another promising avenue for enhancing communication efficiency in FL. This technique, which involves freezing specific model parameters during training, has demonstrated the potential to significantly reduce data transfer without compromising accuracy (Chen et al. 2021a, 2024; Wu et al. 2024).

To address both the computational challenge on edge devices and the necessity for efficient communication in FL, researchers have shifted their attention to methods that could not only simplify the model but also ensure the efficiency of data transmission. Recently, network pruning has been proposed to alleviate the computation costs of deep models(Han, Mao, and Dally 2015; Zhang et al. 2018; Hoefler et al. 2021), and the Lottery Ticket Hypothesis (Frankle and Carbin 2018) suggests that this pruning has minimal, if any, impact on model performance. This finding has spurred significant interest in leveraging model pruning within FL to reduce both computation and communication costs (Li et al. 2020a; Jiang et al. 2022; Bibikar et al. 2022). A notable ex-

*These authors contributed equally.
†Corresponding authors: Tianyun Zhang and Minghai Qin (t.zhang85@csuohio.edu and Minghai.Qin@wdc.com)

ample of such a method is FedDST (Bibikar et al. 2022), which has pioneered a highly influential workflow for FL on extremely resource-constrained edge devices. By dynamically extracting and training sparse sub-networks, FedDST enables clients to work with smaller models, leading to substantial reductions in computation and communication costs. FedDST has served as a foundation for numerous subsequent DST methods (Isik et al. 2022; Babakniya et al. 2023; Huang et al. 2023; Li et al. 2024; Huang et al. 2024). However, these methods typically grant clients more "flexibility" by allowing adjustments to the sparse mask locally. While advantageous in non-independent and identically distributed (non-IID) data settings, this personalized sparse mask training introduces a risk: mask mismatches (Babakniya et al. 2023), which can occur especially under high sparsity, potentially undermining accuracy stability.

In this paper, we introduce **P**arameter **F**reezing-based **F**ederated **D**ynamic **S**parse **T**raining (**PFFDST**), a novel method that inherits the advantages of FedDST while further enhancing efficiency and accuracy in FL environments, particularly on resource-constrained edge devices. While other methods (Isik et al. 2022; Babakniya et al. 2023; Huang et al. 2023; Li et al. 2024; Huang et al. 2024) have built upon FedDST to improve communication efficiency, this work takes a different approach. Specifically, PFFDST leverages parameter freezing, strategically freezing model parameters during sparse training to reduce the communication overhead associated with exchanging model updates, which is particularly beneficial in large-scale deployments. Moreover, PFFDST introduces a novel server-side mask adjustment method to address inconsistencies between locally trained masks and the global model, a prevalent challenge in sparse training for FL. This method readjusts the mask on the server with a differential sparsity, effectively balancing mask stability with the flexibility required for non-IID data settings. Consequently, by synchronizing only the immature parameters in sparse sub-networks instead of full models, PFFDST, combined with server-side mask adjustment, adapts more robustly to the challenges of non-IID data settings and achieves superior performance compared to FedDST. The contributions of this paper are summarized as follows:

1. We propose a novel mask readjustment method for federated dynamic sparse training to improve model accuracy. Different from FedDST which readjusts the mask on clients, our method readjusts masks on the server with a differential sparsity which allows a larger readjustment rate and achieves higher accuracy based on the same communication cost and computation FLOPs.
2. We propose a parameter-freezing method for federated dynamic sparse training to reduce communication costs and computation FLOPs in each round. We approximately reduce communication costs by one-half and computation FLOPs by one-third on average in each round beyond FedDST.
3. Experimental results indicate that each of our proposed methods improves the model accuracy compared with FedDST under the same communication cost and computation FLOPs. When combining our proposed methods, we respectively improve the model accuracy on average by 4% and 6% on ResNet-18 (He et al. 2016) for CIFAR-10 and CIFAR-100 datasets (Krizhevsky, Hinton et al. 2009) compared with FedDST.

Related Work

Federated Learning

Federated Learning is a privacy-preserving approach to machine learning that enables model training across decentralized devices without sharing raw data (Abdulrahman et al. 2021; Aledhari et al. 2020). In FL, a central server manages the global model and periodically communicates with clients. Clients download the latest model parameters, perform local training over multiple iterations, and then upload their updates for global aggregation. FedAvg(McMahan et al. 2017) is the most common global aggregation method, which simply averages the model updates uploaded to the server. System and statistical heterogeneity across clients is a key challenge in FL. Methods like FedProx (Li et al. 2020b) and FedNova (Wang et al. 2020) achieve more stable and accurate convergence compared to FedAvg in highly heterogeneous settings.

Besides the challenge of heterogeneity, communication efficiency is another critical challenge in FL (Konečnỳ et al. 2016; Chen et al. 2021b), particularly for resource-constrained edge devices. To enhance communication efficiency in FL, one straightforward strategy involves compressing model updates using methods like sparsification (Ozfatura, Ozfatura, and Gündüz 2021; Isik, Weissman, and No 2022), quantization (Bernstein et al. 2018; Mitchell et al. 2022), and low-rank approximation (Mohtashami, Jaggi, and Stich 2022; Wang et al. 2018). However, these methods still require training a dense network, which may not be deployable on edge devices with limited memory and computational resources.

Parameter-Freezing for FL

Parameter-freezing is a technique in machine learning that can improve efficiency without significantly compromising performance. It involves freezing specific model parameters at their initial values during training, thereby reducing computational and storage costs (Wimmer, Mehnert, and Condurache 2020). This technique has recently shown great potential for enhancing communication efficiency in FL. For instance, Chen et al. (Chen et al. 2021a, 2024) proposed Adaptive parameter-freezing (APF), which identifies stable parameters and excludes them from synchronization, reducing data transfer by over 60% without compromising model accuracy. Similarly, Wu et al. (Wu et al. 2024) extended this concept to federated vehicular networks, combining parameter-freezing with bandwidth allocation optimization to minimize communication overhead and latency. These methods effectively address the challenge of limited network bandwidth in edge devices by selectively synchronizing only necessary parameters.

However, these existing works primarily focus on applying parameter-freezing to dense networks. The potential

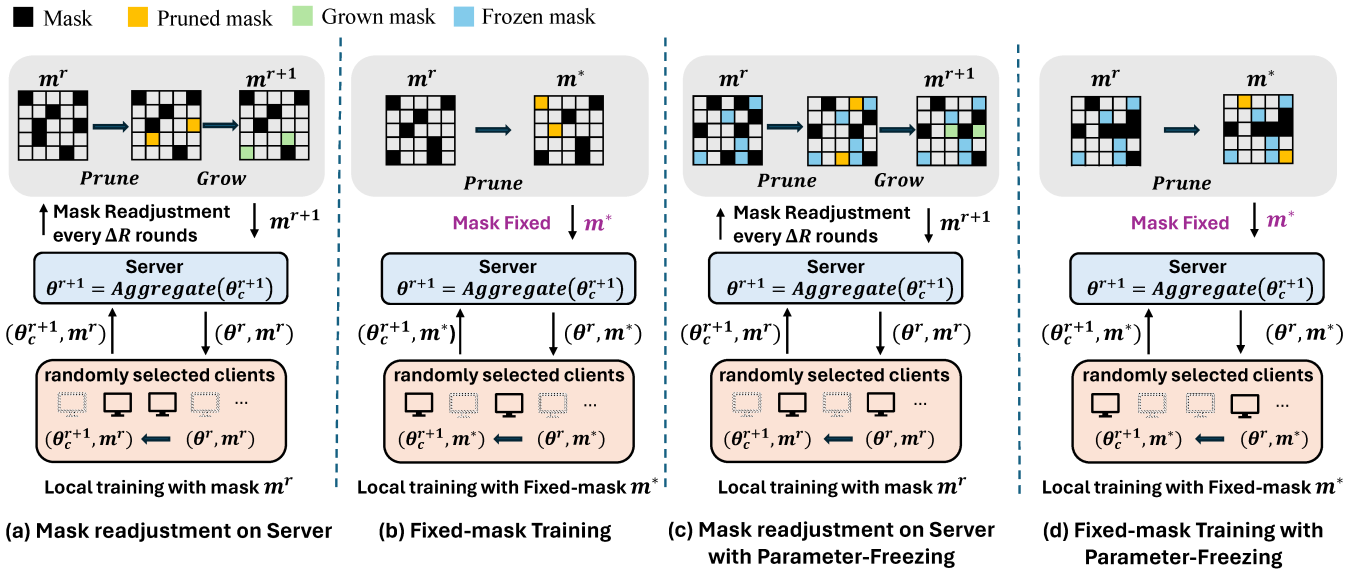


Figure 1: Overview of the PFFDST framework.

of leveraging this technique within the context of dynamic sparse training remains largely unexplored. This paper addresses this gap by proposing a novel approach that integrates parameter-freezing into the dynamic sparse training process. Unlike existing methods, our approach freezes both the sparse mask and the corresponding weights, leading to two key advantages: (1) direct control over the model's sparsity structure, ensuring a consistently efficient representation throughout training, and (2) a more synergistic optimization process that can further reduce communication costs without sacrificing accuracy.

Dynamic Sparse Training for FL

The introduction of the Lottery Ticket Hypothesis (LTH) (Frankle and Carbin 2018), which posits that a sparse network can be found at initialization, has proved the feasibility of identifying effective sparse architectures from the onset of training. The original paper proposed finding these sparse networks by iteratively training the dense network using pruning techniques (Han, Mao, and Dally 2015; Zhang et al. 2018; Hoefler et al. 2021). This process involves continuously removing unimportant weights and reassigning the values of the remaining weights, ultimately yielding lightweight neural networks with a low memory footprint and reduced communication overhead. These characteristics make them well-suited for deployment on edge devices, particularly in resource-constrained and bandwidth-restricted FL scenarios.

Dynamic sparse training (DST) further advances this concept as a dynamic process that extracts and trains sparse subnetworks from the target full network (Mostafa and Wang 2019; Liu et al. 2020). Early works like LotteryFL (Li et al. 2020a) and PruneFL (Jiang et al. 2022) aimed to improve communication efficiency in FL through DST. While both LotteryFL and PruneFL suffer from the need to frequently communicate large amounts of model information.

FedDST (Bibikar et al. 2022), a widely adopted FL method, addresses these limitations by dynamically extracting and training sparse sub-networks and communicating only these sub-networks to generate a globally optimal sparse model, significantly reducing communication costs. Building on this foundation, researchers have investigated techniques for improved initialization, convergence, and efficiency. For instance, FLASH (Babakniya et al. 2023) and FedTiny (Huang et al. 2023) focus on finding better initial sparse masks before FL training, with FLASH also enabling layer-wise sparsity for enhanced stability. DSFedCon (Li et al. 2024) tackles performance by introducing a contrastive loss, while FedMef (Huang et al. 2024) addresses memory efficiency with DROPIT (Chen et al. 2022). Another work (Isik et al. 2022) achieves a 32× reduction in communication by transmitting only binary mask updates. However, this method demands significant client-side computation, which is beyond the scope of this work.

Methodology

Figure 1 shows the overview of our proposed framework which includes four stages. The first stage, Mask Readjustment on Server, involves adjusting the optimal sparse mask on the server every ΔR communication rounds through a process of top-k pruning and random growth. The second stage, Fixed Mask Training, involves training the global model with the previously optimized sparse mask fixed, which helps to achieve faster and more stable convergence. The third and fourth stages are the parameter-freezing stages, which start with freezing the previously learned optimal parameters and mask, followed by randomly growing the new mask and weights. Similar to the first stage, the third stage (Figure 1(c)) readjusts the newly grown mask, and the fourth stage (Figure 1(d)) fixes the final mask and then continues training. This methodological approach aims

Algorithm 1: Mask readjustment on the server with differential sparsity

Input: Clients $[N]$ with local datasets, target sparsity s_1, differential sparsity f_1, ΔR, R_{end}, total rounds R, number of epochs E in each round.
Initialize server model (θ^1, m^1) at sparsity $(s_1 - f_1)$.
for $r = 1, 2, \ldots, R$ **do**
  Sample clients $C_r \subset [N]$.
  for each client $c \in C_r$ **do**
    Transmit the server model (θ^r, m^r) to client c.
    Perform local training with mask m^r for E epochs.
    Transmit the updated model parameters θ_c^{r+1} from client c to the server.
  end for
  Aggregate the received parameters on the server to derive θ^{r+1}, $m^{r+1} = m^r$.
  if r mod ΔR = 0 and $r < R_{end}$ **then**
    $(\theta^{r+1}, m^{r+1}) \leftarrow TopkPrune(\theta^{r+1}, s_1)$.
    $(\theta^{r+1}, m^{r+1}) \leftarrow RandomGrow(\theta^{r+1}, s_1 - f_1)$.
  else if $r = R_{end}$ **then**
    $(\theta^{r+1}, m^{r+1}) \leftarrow TopkPrune(\theta^{r+1}, s_1)$.
  end if
end for

to refine and stabilize the learning process within federated sparse training through strategic mask adjustments.

Proposed Framework

We first initialize a sparse network with parameters θ^1 and a sparse mask m^1 on the server, which follows FedDST (Bibikar et al. 2022). Unlike (Bibikar et al. 2022), which readjusts the masks on the clients, we readjust the masks on the server. Assuming the target sparsity is s_1, we initialize a network with sparsity $(s_1 - f_1)$, where f_1 denotes the differential sparsity which is the difference between the sparsity of the grown model and target sparsity. In each round, we first sample clients and transmit model parameters and masks to them. After that, the model is trained in the sampled clients with the masks for E epochs. Thirdly, the model parameters are transmitted from the sampled clients to the server. Finally, we update (θ^r, m^r) in different ways based on the condition of the current round. We readjust the masks on the server every ΔR rounds during the first R_{end} rounds. When readjusting masks, we first perform $TopkPrune(\cdot)$ to prune the parameters to sparsity s_1 based on their magnitude. Then we perform $RandomGrow(\cdot)$ to randomly grow the parameters, adjusting the network's sparsity to $(s_1 - f_1)$. At the R_{end} round, we prune the parameters to sparsity s_1, with the mask remaining unchanged. The details of our proposed DST method on FL are summarized in Algorithm 1.

We also propose a two-step parameter-freezing method to reduce the communication costs in FL. The details of the proposed method are summarized in Algorithm 2. Firstly, we apply Algorithm 1 to train a model $(\theta, \hat{m})$ with sparsity s_1. Secondly, we freeze all the non-zero parameters in θ and train a model with target sparsity s_2 and differential sparsity f_2, and then randomly grow the model from sparsity s_1 to

Algorithm 2: Parameter-freezing-based dynamic sparse training

Input: Clients with local datasets, target sparsities s_1 and s_2, differential sparsities f_1 and f_2, ΔR, R_{end}, total rounds $2R$, number of epochs E in each round.
Input s_1, f_1, ΔR, R_{end}, total round R, and E to Algorithm 1 to train a model $(\theta, \hat{m})$.
Freeze the non-zero parameters in θ and randomly grow it to the sparsity $(s_2 - f_2)$ on the server, which derives the model (θ^{R+1}, m^{R+1}).
for $r = R+1, R+2, \ldots, 2R$ **do**
  Sample clients $C_r \subset [N]$.
  for each client $c \in C_r$ **do**
    Transmit $(\theta^r \circ (1 - \hat{m}), m^r)$ to client c. Set the local model to $\theta_c^r = \hat{\theta}_c + \theta^r \circ (1 - \hat{m})$.
    Perform local training with mask $m^r \circ (1 - \hat{m})$ for E epochs.
    Transmit the updated model parameters $\theta_c^{r+1} \circ (1 - \hat{m})$ from client c to the server.
  end for
  Aggregate the received parameters on the server to derive $\bar{\theta}^{r+1}$, $\theta^{r+1} = \bar{\theta}^{r+1} + \theta$,$m^{r+1} = m^r$.
  if $(r - R)$ mod ΔR = 0 and $r < (R + R_{end})$ **then**
    $(\theta^{r+1}, m^{r+1}) \leftarrow TopkPrune(\theta^{r+1}, s_2)$.
    $(\theta^{r+1}, m^{r+1}) \leftarrow RandomGrow(\theta^{r+1}, s_2 - f_2)$.
  else if $r = (R + R_{end})$ **then**
    $(\theta^{r+1}, m^{r+1}) \leftarrow TopkPrune(\theta^{r+1}, s_2)$.
  end if
end for

$(s_2 - f_2)$. In each round, we first sample clients and transmit the model parameters and masks to them. If a client has not been sampled after the R-th round, we need to transmit both the server model and the mask of the frozen parameters $\hat{m}$ to the client. Then the frozen parameters will be saved in this client. If this client is sampled again in a later round, we only need to transmit the unfrozen parameters $\theta^r \circ (1 - \hat{m})$ to the client, which significantly reduces the communication costs. Specifically, the nonzero parameters in θ^r are $(1 - s_2)$ fraction of the total parameters, in which $(1 - s_1)$ of the total parameters have been frozen. Thus only $(s_1 - s_2)$ fraction of the total parameters is necessary to be transmitted between the server and the clients. From round $(R+1)$ to round $(R + R_{end})$, we readjust the mask every ΔR rounds. At the round $(R + R_{end})$, we prune the parameters to sparsity s_2 with the mask remaining unchanged finally.

Discussions on the Proposed Mask Readjustment-on-Server Method

As discussed in FedDST (Bibikar et al. 2022), when mask readjustment is performed on clients, each client submits a "vote" to the server regarding the mask readjustment. The server then determines how to adjust the mask based on the collected "votes". For this method, FedDST suggests setting the mask readjustment ratio in the range of 0.001 to 0.05, which is much smaller than that of regular DST. A larger readjustment ratio will prevent the mask on the server from

converging to an optimal state, thereby reducing the accuracy of the sparse model. However, we find that setting the mask readjustment ratio in the range of 0.001 to 0.05 cannot significantly improve model accuracy compared with the RandomMask method in which the mask readjustment ratio is 0. Experimental results in (Bibikar et al. 2022) illustrate that FedDST usually can only improve the accuracy by 1% based on the RandomMask method.

To further improve the model accuracy achieved by FedDST, we propose the framework to readjust the mask on the server instead of clients. This approach has been explored in several earlier works (Jiang et al. 2022; Huang et al. 2023). However, these methods fail to address the challenges of non-IID data, particularly the incomparability of gradients across clients in non-IID environments.

In this case, we set a higher mask readjustment ratio than that of FedDST. When we set the target sparsity to s and the differential sparsity to f, the readjustment ratio is given by $\alpha = \frac{f}{1-s}$. As the server does not contain the gradient of the aggregated model, we choose to randomly grow the model from sparsity s to $(s-f)$, transmitting the model with sparsity $(s-f)$ to the clients for training and pruning the updated aggregated model back to sparsity s after ΔR rounds. In our method, since we include a differential sparsity f, we increase the communication cost and computation FLOPs by $\alpha\times$ in the rounds that we readjust the masks (rounds 1 to R_{end} and rounds $R+1$ to $R+R_{end}$). As we suggest setting the readjustment ratio to 0.5, this increase will be 50% in each round when we readjust the masks. However, we only need a quarter of the total rounds for mask readjustment by setting $R_{end} = \frac{R}{4}$, thus the differential sparsity will not highly affect the overall communication cost and computation FLOPs. When we reduce the total number of rounds while maintaining the same overall communication cost and computation FLOPs as other methods, we achieve significantly higher model accuracy.

Discussions on the Proposed Parameter-Freezing Method

Communication Analysis. Assume the target sparsity is s_1 and s_2 in our first and second steps, respectively, where $s_1 > s_2$ and both s_1 and s_2 are between 0 and 1. The communication cost in each round is approximately $(1-s_1)$ and (s_1-s_2) fraction of the dense model in the first and second steps, respectively (the exact cost can be affected by the differential sparsity, mask, and biases). For the sparse training methods without parameter-freezing, the communication cost in each round is approximately $(1-s_2)$ of the dense model when the target sparsity is s_2. In practice, we set $s_1 = \frac{1+s_2}{2}$, thus that $1-s_1 = s_1-s_2 = \frac{1-s_2}{2}$. The communication cost in each round is approximately one-half of a sparse training method without the parameter-freezing method.

Computation Analysis. The computation in training a DNN is composed of three phases, namely, forward propagation, backward propagation, and parameter gradient computation. These three phases have equal FLOPs in theory. In our first step, the FLOPs are $(1-s_1)$ fraction of training the dense model for all these three phases. In our second step, the FLOPs are $(1-s_2)$ of the dense model for forward propagation and backward propagation phase. Our FLOPs are only (s_1-s_2) of the dense model for parameter gradient computation phase since we freeze the parameters obtained in the first step. In practice, we set $s_1 = \frac{1+s_2}{2}$ and we use the same number of rounds in the first and second steps. Therefore, the average FLOPs per round are $\frac{1}{2}(1-s_1) + \frac{1}{2}(\frac{2}{3}(1-s_2) + \frac{1}{3}(s_1-s_2)) = \frac{2}{3}(1-s_2)$ of the dense model. For sparse training without parameter-freezing and target sparsity being s_2, the training FLOPs are $(1-s_2)$ fraction of the dense model. It can be observed that parameter-freezing reduces the training FLOPs by one-third with the same target sparsity s_2.

Commu. Cost (GB)	4	8	12	16
Tra. FLOPs ($\times 10^{13}$)	0.23	0.46	0.69	0.92
FedAvgM (dense)	24.43	33.87	37.07	40.52
FedProx (dense)	23.54	34.01	39.08	42.56
FedDST	35.41	42.27	46.72	50.67
FedDST+FedProx	33.03	43.18	46.66	49.69
PFFDST (ours)	**39.21**	**47.35**	**48.55**	**55.87**

Table 1: Experiment results of PFFDST compared with other dense and sparse methods given cumulative bandwidth limits, on non-IID CIFAR-10 using LeNet. We set $s = 0.8$.

Experiments

Experimental Setups

This paper evaluates the performance of a proposed framework, PFFDST[1], against established FL techniques on CIFAR-10 (Krizhevsky, Hinton et al. 2009) and CIFAR-100 (Krizhevsky, Hinton et al. 2009) datasets using LeNet (LeCun et al. 1998) and ResNet-18 (He et al. 2016) models. The datasets are partitioned across multiple clients using a Dirichlet distribution ($\alpha = 0.1$) to simulate non-IID data settings. LeNet experiments are evaluated based on the highest accuracy, aligning with established FedDST (Bibikar et al. 2022) benchmarks. In the experiments using ResNet-18, we report the average accuracy with error bounds to provide a comprehensive performance assessment. These error bounds, derived from multiple experimental runs, offer a robust measure of the variability in performance and enhance the statistical significance of the reported average accuracy.

Specifically, PFFDST is compared against several dense baselines (FedAvg (McMahan et al. 2017), FedProx (Li et al. 2020b)) and sparse baselines (FedDST (Bibikar et al. 2022), FedDST+FedProx (Bibikar et al. 2022)). We conduct experiments with varying target sparsities (s). For LeNet, we fix $s = 0.8$, while for ResNet-18, we explore a range of sparsities: $s \in 0.9, 0.95, 0.98, 0.99$. PFFDST configurations utilize two sparsity levels: $s_1 = (s+1)/2$ and $s_2 = s$,

[1]Code and Appendix: https://github.com/Dawns14/pffdst.git

Commu. Cost (GB)	34	68	102	136	17	34	51	68
Tra. FLOPs ($\times 10^{13}$)	16	32	48	64	8	16	24	32
FedAvgM (dense)	29.76±1.7	42.86±1.7	48.60±2.2	51.49±2.2	21.74±1.4	29.76±1.7	37.62±2.2	42.86±1.7
FedProx(dense)	31.17±2.1	40.81±1.5	50.89±2.8	53.25±2.2	21.52±2.9	31.17±2.1	35.94±2.1	40.81±1.5
	s = 0.9				s = 0.95			
FedDST	53.41±1.8	58.92±1.4	58.96±1.5	60.51±1.4	51.92±1.1	54.69±1.3	56.18±1.5	56.83±1.8
FedDST+FedProx	51.70±1.9	56.05±0.6	58.64±0.8	60.70±1.7	51.07±1.5	54.98±2.2	55.84±1.9	57.49±1.2
PFFDST (ours)	**57.81**±3.1	**61.59**±0.4	**64.01**±0.4	**64.18**±0.5	**56.94**±1.3	**59.16**±1.3	**60.03**±0.7	**61.89**±0.5

Table 2: Experiment results of PFFDST compared with other dense and sparse methods given cumulative bandwidth limits, on non-IID CIFAR-10 using ResNet-18. We set $s = 0.9, 0.95$.

Commu. Cost (GB)	34	68	102	136	17	34	51	68
Train. FLOPs ($\times 10^{13}$)	16	32	48	64	8	16	24	32
FedAvgM (dense)	20.20±0.4	26.76±0.8	29.46±0.4	32.56±0.7	13.94±0.4	20.20±0.4	24.58±0.7	26.76±0.8
FedProx(dense)	20.42±0.8	25.83±0.2	29.17±0.5	31.44±0.6	14.47±0.9	20.42±0.8	23.68±0.5	25.83±0.2
	s = 0.9				s = 0.95			
FedDST	31.31±0.4	33.67±0.3	34.18±0.5	34.90±0.4	30.28±0.2	31.91±0.4	32.96±0.2	33.17±0.5
FedDST+FedProx	31.42±0.5	33.48±0.6	34.23±0.5	34.78±0.9	30.60±0.5	32.00±0.3	32.91±0.2	33.16±0.3
PFFDST (ours)	**38.85**±0.1	**39.47**±0.2	**40.51**±0.3	**40.55**±0.3	**37.16**±0.6	**37.92**±0.6	**38.96**±0.1	**40.08**±1.1

Table 3: Experiment results of PFFDST compared with other dense and sparse methods given cumulative bandwidth limits, on non-IID CIFAR-100 using ResNet-18. We set $s = 0.9, 0.95$.

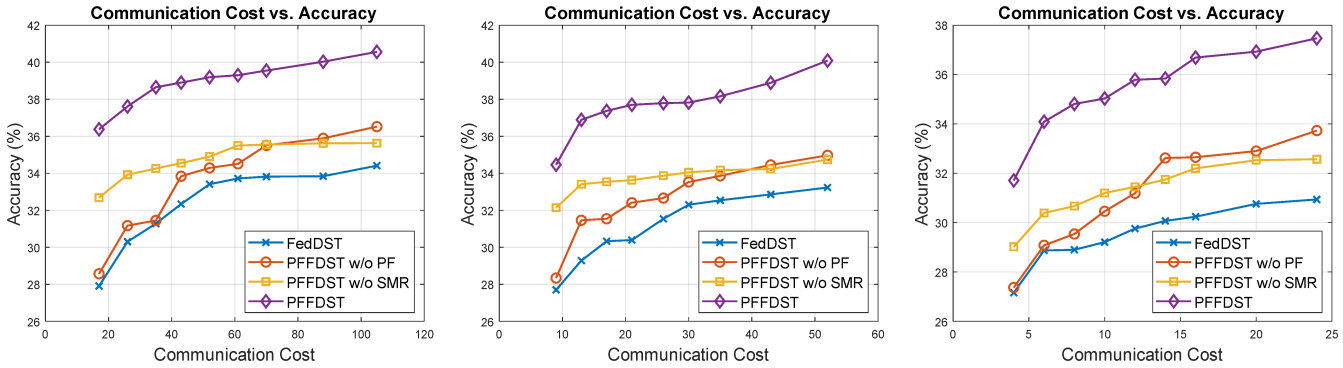
with differential sparsities $f_1 = f_2 = (1 - s)/4$ to control communication overhead. The number of training epochs is set to 3 for FedDST and 2 for PFFDST to maintain comparable FLOPs. Additional parameters include $\Delta R = 10$, $R_{end} =$ total rounds$/8$, 400 clients for LeNet, 200 clients for ResNet-18, and 20 randomly selected clients per communication round. The implementation leverages PyTorch (Paszke et al. 2019) on a server equipped with 8 A6000 GPUs, with detailed hyperparameter settings outlined in Appendix A.

Performance Evaluation

PFFDST achieves superior performance compared to baseline methods on the CIFAR-10 dataset using the LeNet network, as shown in Table 1. Following the methodology of FedDST (Bibikar et al. 2022), we report the highest testing accuracy achieved across 10 runs with a fixed sparsity level of 0.8 (as in FedDST). Notably, PFFDST achieves an average accuracy improvement of 4% over the FedDST and FedDST+FedProx method, even under low communication cost and training FLOPs constraints.

To further evaluate the performance of PFFDST, we extend our analysis to larger and more complex networks, such as ResNet-18. Considering the large number of parameters and the complexity of the ResNet-18 network, multiple sparsity levels significantly affect the network's trainable parameters. Therefore, we compare the performance of PFFDST under different sparsity levels based on the ResNet-18 network. For model accuracy, we repeat each experiment 7 times, discard the highest and lowest values, and report the mean and standard deviation based on the remaining 5 results on CIFAR-10 and CIFAR-100 datasets for both methods. Table 2 shows the results of applying sparsity ratios $s = 0.9, 0.95$ on the CIFAR-10 dataset based on the ResNet-18 network. Our proposed PFFDST method shows significant performance improvement under multiple data transmission constraints. The most notable improvement is observed at a data transmission limit of 17GB and Training FLOPs of $8 * 10^{13}$, with an accuracy increase of 5%. On the more challenging CIFAR-100 dataset, which has a greater number of categories, the results in Table 3 further emphasize the effectiveness of PFFDST, with an average accuracy increase of 6%. We perform additional experiments under extreme sparsity settings ($s = 0.98, 0.99$), with the results presented in Appendix B. Furthermore, we extend our analysis by comparing PFFDST with other recent competitive methods, such as FedTiny (Huang et al. 2023), FedMef (Huang et al. 2024), and FLASH (Babakniya et al. 2023), and evaluating its performance on advanced transformer-based models. More detailed experimental results are provided in Appendix C.

Overall, our analysis indicates that PFFDST can adapt to various sparsity levels, consistently showing performance improvements. Additionally, when targeting the same accu-



(a) s = 0.90 (b) s = 0.95 (c) s = 0.98

Figure 2: Comparison of the accuracy of PFFDST and its variants at three different sparsity levels $s = \{0.90, 0.95, 0.98\}$ with varying communication costs on non-IID CIFAR-100 using ResNet-18.

racy, the proposed method can reduce communication costs by 4-8 times for different datasets with varying sparsity levels. This adaptability is crucial for real-world deployments, where significant variations in computational resources and data availability may occur.

Ablation Study

In previous experiments, the exceptional performance of PFFDST was fully verified. However, PFFDST combines two distinct techniques: mask readjustment on the server with differential sparsity and parameter-freezing. This raises the question of which method plays a decisive role. To answer this, we conducted an ablation study on PFFDST. We compared the baseline FedDST with its three variants: (1) PFFDST without parameter-freezing (PFFDST w/o PF), which incorporates only mask readjustment on the server with differential sparsity; (2) PFFDST without server mask readjustment (PFFDST w/o SMR), which incorporates only parameter-freezing; and (3) the fully integrated version of PFFDST. We conducted our ablation experiments on the CIFAR-100 dataset using a ResNet-18 network to validate the results. Furthermore, the impact of the readjustment ratio is analyzed in Appendix D.

The impacts between PFFDST w/o PF and PFFDST w/o SMR. Both PFFDST w/o PF and PFFDST w/o SMR outperform FedDST in accuracy under all communication cost constraints, showing the effectiveness of each strategy. Under a 4GB communication limit (Figure 2(c)), PFFDST w/o PF improves accuracy by 0.2%, whereas PFFDST w/o SMR achieves a 2% gain, highlighting the dominant role of parameter-freezing. At a 24GB communication limit, PFFDST w/o PF improves accuracy by 2.8% over FedDST, while PFFDST w/o SMR achieves a 1.6% gain. This reversal suggests that mask readjustment with differential sparsity becomes increasingly important as communication capacity increases. Figure 2(a) and (b) present results for different sparsity levels.

The impacts combining PFFDST w/o PF and PFFDST w/o SMR. When PFFDST combines both strategies, it improves on average by 6% compared to FedDST. This improvement is significantly greater than that achieved by methods that apply only one of the strategies. Overall, the results highlight the effectiveness of combining these two techniques within the PFFDST framework, especially in scenarios with constrained communication costs. These complementary strategies work synergistically to enhance accuracy, showcasing PFFDST's adaptability across diverse communication environments.

Conclusion

In this paper, we propose Parameter-Freezing-based Federated Dynamic Sparse Training (PFFDST), a framework that can achieve higher model accuracy in federated learning with lower communication cost and computation FLOPs. The first method in our proposed framework is a novel sparse mask readjustment rule and the second is a parameter-freezing method during training. These two methods complement each other, leading to a synergistic effect that further enhances the performance of PFFDST. Experimental results show that applying each method individually yields better performance than FedDST. When we combine both methods, PFFDST achieves significantly better performance than FedDST. Specifically, we improve the model accuracy on average by 4% and 6% on ResNet-18 for CIFAR10 and CIFAR-100 respectively compared with FedDST under the same communication cost and computation FLOPs. Large language models represent a significant trend in artificial intelligence, showcasing superior capabilities in diverse tasks, but their training in federated learning on resource-limited devices faces significant computational and communication challenges. Future work can explore the potential of PFFDST in federated large language models to facilitate broader real-world applications.

Acknowledgements

This research was supported by the National Science Foundation under award CNS-2245765.

References

Abdulrahman, S.; Tout, H.; Ould-Slimane, H.; Mourad, A.; Talhi, C.; and Guizani, M. 2021. A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond. *IEEE Internet of Things Journal*, 8(7): 5476–5497.

Albasyoni, A.; Safaryan, M.; Condat, L.; and Richtárik, P. 2020. Optimal gradient compression for distributed and federated learning. *arXiv preprint arXiv:2010.03246*.

Aledhari, M.; Razzak, R.; Parizi, R. M.; and Saeed, F. 2020. Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Access*, 8: 140699–140725.

Babakniya, S.; Kundu, S.; Prakash, S.; Niu, Y.; and Avestimehr, S. 2023. Revisiting Sparsity Hunting in Federated Learning: Why does Sparsity Consensus Matter? *Transactions on Machine Learning Research*.

Bernstein, J.; Wang, Y.-X.; Azizzadenesheli, K.; and Anandkumar, A. 2018. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, 560–569. PMLR.

Bibikar, S.; Vikalo, H.; Wang, Z.; and Chen, X. 2022. Federated dynamic sparse training: Computing less, communicating less, yet learning better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 6080–6088.

Chen, C.; Xu, H.; Wang, W.; Li, B.; Li, B.; Chen, L.; and Zhang, G. 2021a. Communication-Efficient Federated Learning with Adaptive Parameter Freezing. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, 1–11.

Chen, C.; Xu, H.; Wang, W.; Li, B.; Li, B.; Chen, L.; and Zhang, G. 2024. Synchronize Only the Immature Parameters: Communication-Efficient Federated Learning By Freezing Parameters Adaptively. *IEEE Transactions on Parallel and Distributed Systems*, 35(7): 1155–1173.

Chen, J.; Xu, K.; Wang, Y.; Cheng, Y.; and Yao, A. 2022. Dropit: Dropping intermediate tensors for memory-efficient dnn training. *arXiv preprint arXiv:2202.13808*.

Chen, M.; Shlezinger, N.; Poor, H. V.; Eldar, Y. C.; and Cui, S. 2021b. Communication-efficient federated learning. *Proceedings of the National Academy of Sciences*, 118(17): e2024789118.

Chen, T.; Frankle, J.; Chang, S.; Liu, S.; Zhang, Y.; Carbin, M.; and Wang, Z. 2021c. The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 16306–16316.

Chen, T.; Frankle, J.; Chang, S.; Liu, S.; Zhang, Y.; Wang, Z.; and Carbin, M. 2020. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33: 15834–15846.

Frankle, J.; and Carbin, M. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Gale, T.; Elsen, E.; and Hooker, S. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hoefler, T.; Alistarh, D.; Ben-Nun, T.; Dryden, N.; and Peste, A. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241): 1–124.

Huang, H.; Zhang, L.; Sun, C.; Fang, R.; Yuan, X.; and Wu, D. 2023. Distributed pruning towards tiny neural networks in federated learning. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, 190–201. IEEE.

Huang, H.; Zhuang, W.; Chen, C.; and Lyu, L. 2024. FedMef: Towards Memory-efficient Federated Dynamic Pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 27548–27557.

Isik, B.; Pase, F.; Gunduz, D.; Weissman, T.; and Zorzi, M. 2022. Sparse random networks for communication-efficient federated learning. *arXiv preprint arXiv:2209.15328*.

Isik, B.; Weissman, T.; and No, A. 2022. An information-theoretic justification for model pruning. In *International Conference on Artificial Intelligence and Statistics*, 3821–3846. PMLR.

Jiang, Y.; Wang, S.; Valls, V.; Ko, B. J.; Lee, W.-H.; Leung, K. K.; and Tassiulas, L. 2022. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12): 10374–10386.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2): 1–210.

Khan, L.; et al. 2021. Privacy and Security Challenges in the Internet of Things (IoT). In *Proceedings of the 2021 Conference on Internet of Things*.

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

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, A.; Sun, J.; Wang, B.; Duan, L.; Li, S.; Chen, Y.; and Li, H. 2020a. Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets. *arXiv preprint arXiv:2008.03371*.

Li, H.; Ota, K.; and Dong, M. 2018. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. *IEEE network*, 32(1): 96–101.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020b. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2: 429–450.

Li, Z.; Chen, J.; Zhang, P.; Huang, H.; and Li, G. 2024. DS-FedCon: Dynamic Sparse Federated Contrastive Learning for Data-Driven Intelligent Systems. *IEEE Transactions on Neural Networks and Learning Systems*.

Lin, Y.; Han, S.; Mao, H.; Wang, Y.; and Dally, B. 2018. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *International Conference on Learning Representations*.

Liu, J.; Xu, Z.; Shi, R.; Cheung, R. C.; and So, H. K. 2020. Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers. In *International Conference on Learning Representations*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Mitchell, N.; Ballé, J.; Charles, Z.; and Konečnỳ, J. 2022. Optimizing the communication-accuracy trade-off in federated learning with rate-distortion theory. *arXiv preprint arXiv:2201.02664*.

Mohammadi, M.; Al-Fuqaha, A.; Sorour, S.; and Guizani, M. 2018. Deep learning for IoT big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 20(4): 2923–2960.

Mohtashami, A.; Jaggi, M.; and Stich, S. 2022. Masked training of neural networks with partial gradients. In *International Conference on Artificial Intelligence and Statistics*, 5876–5890. PMLR.

Mostafa, H.; and Wang, X. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, 4646–4655. PMLR.

Nguyen, D.; et al. 2021. Data Privacy in IoT: Challenges and Solutions. In *Proceedings of the 2021 Workshop on IoT Security*.

Ozfatura, E.; Ozfatura, K.; and Gündüz, D. 2021. Time-correlated sparsification for communication-efficient federated learning. In *2021 IEEE International Symposium on Information Theory (ISIT)*, 461–466. IEEE.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Vogels, T.; Karimireddy, S. P.; and Jaggi, M. 2019. PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Voigt, P.; and Von dem Bussche, A. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676): 10–5555.

Wang, H.; Sievert, S.; Liu, S.; Charles, Z.; Papailiopoulos, D.; and Wright, S. 2018. Atomo: Communication-efficient learning via atomic sparsification. *Advances in neural information processing systems*, 31.

Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; and Poor, H. V. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33: 7611–7623.

Wimmer, P.; Mehnert, J.; and Condurache, A. 2020. FreezeNet: Full Performance by Reduced Storage Costs. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*.

Wu, J.; Dai, T.; Guan, P.; Liu, S.; Gou, F.; Taherkordi, A.; Li, Y.; and Li, T. 2024. FedAPT: Joint Adaptive Parameter Freezing and Resource Allocation for Communication-Efficient Federated Vehicular Networks. *IEEE Internet of Things Journal*, 11(11): 19520–19536.

Zhang, C.; Patras, P.; and Haddadi, H. 2019. Deep learning in mobile and wireless networking: A survey. *IEEE Communications surveys & tutorials*, 21(3): 2224–2287.

Zhang, T.; Ye, S.; Zhang, K.; Tang, J.; Wen, W.; Fardad, M.; and Wang, Y. 2018. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European conference on computer vision (ECCV)*, 184–199.