Federated Dynamic Sparse Training: Computing Less, Communicating Less, Yet Learning Better

Sameer Bibikar, Haris Vikalo, Zhangyang Wang, Xiaohan Chen[*]

Department of Electrical and Computer Engineering, The University of Texas at Austin
{bibikar,hvikalo,atlaswang,xiaohan.chen}@utexas.edu

Abstract

Federated learning (FL) enables distribution of machine learning workloads from the cloud to resource-limited edge devices. Unfortunately, current deep networks remain not only too compute-heavy for inference and training on edge devices, but also too large for communicating updates over bandwidth-constrained networks. In this paper, we develop, implement, and experimentally validate a novel FL framework termed *Federated Dynamic Sparse Training* (**FedDST**) by which complex neural networks can be deployed and trained with substantially improved efficiency in both **on-device computation** and **in-network communication**. At the core of FedDST is a dynamic process that extracts and trains sparse sub-networks from the target full network. With this scheme, "two birds are killed with one stone:" instead of full models, each client performs efficient training of its own sparse networks, and only sparse networks are transmitted between devices and the cloud. Furthermore, our results reveal that the dynamic sparsity during FL training more flexibly accommodates local heterogeneity in FL agents than the fixed, shared sparse masks. Moreover, dynamic sparsity naturally introduces an "in-time self-ensembling effect" into the training dynamics, and improves the FL performance **even over dense training**. In a realistic and challenging non i.i.d. FL setting, FedDST consistently outperforms competing algorithms in our experiments: for instance, at any fixed upload data cap on non-iid CIFAR-10, it gains an impressive accuracy advantage of 10% over FedAvgM when given the same upload data cap; the accuracy gap remains 3% even when FedAvgM is given 2× the upload data cap, further demonstrating efficacy of FedDST. Code is available at: https://github.com/bibikar/feddst.

Introduction

Driven by the desire to protect the privacy of personal data and enable machine learning (ML) at the edge, Federated Learning (FL) (McMahan et al. 2017; Kairouz et al. 2019) has recently emerged as the *de facto* paradigm enabling distributed ML on a large number of client devices. In a FL system, a central cloud server mediates information transfer within a network of clients which must keep their local data private. Classical methods (McMahan et al. 2017) in FL involve a number of synchronous *rounds*; in each round, FL runs several local training epochs on a subset of devices using only data available locally on each device. After this local training, the clients' model updates, rather than the local data, are sent over to the central server which then aggregates them all to update a global model.

In FL systems, heavy computational workloads are dispatched from the cloud to resource-limited edge devices. To enable usage at the edge, a FL system must optimize both device-level local training efficiency and in-network communication efficiency. Unfortunately, current ML models are typically too complex for inference at edge devices, not to mention training. Besides model compactness, communication efficiency between the cloud and devices is also desirable. Client devices, such as mobile phones, often have severe upload bandwidth limitations due to asymmetric internet connections, so reducing the upload cost of federated learning algorithms is of paramount importance. Much prior work in communication-efficient FL has focused on structured and sketched sparsity in FL updates (Konečný et al. 2017), optimal client sampling (Ribero and Vikalo 2020), and other classical methods.

With the goal of producing lightweight models for inference on edge devices, significant efforts have been made towards optimizing sparse neural networks (NNs) (Gale, Elsen, and Hooker 2019; Chen et al. 2020, 2021; Ma et al. 2021). These methods significantly reduce inference latency, but heavily impact compute and memory resources needed for training. The lottery ticket hypothesis (Frankle and Carbin 2018) demonstrated that dense NNs contain sparse matching subnetworks that are capable of training in isolation to full accuracy (Frankle et al. 2020). More works show sparsity can emerge at initialization (Lee, Ajanthan, and Torr 2019; Wang, Zhang, and Grosse 2020) or can be exploited in dynamic forms during training (Evci et al. 2020).

The overarching goal of this paper is to develop, implement, and experimentally validate a novel FL framework termed *Federated Dynamic Sparse Training* (**FedDST**), by which complex NNs can be deployed and trained with substantially improved efficiency of both on-device computation and in-network communication. At the core of FedDST is a judiciously designed federated approach to dynamic sparse training (Evci et al. 2020). FedDST transmits

[*]Xiaohan Chen is the corresponding author, and one of the two student authors who made major contributions in this work.

clients' highly sparse matching subnetworks instead of full models, and allows each client to plug in efficient sparse distributed training - thus "killing two birds with one stone." More importantly, we discover that dynamic sparsity during FL training accommodates local heterogeneity in FL more robustly than state-of-the-art algorithms. Dynamic sparsity itself leads to an in-time self-ensembling effect (Liu et al. 2021c) and improves the FL performance even over dense training counterparts, which echoes observations in standalone training (Liu et al. 2021c). We summarize our contributions as follows:

- For the first time, we introduce dynamic sparse training to federated learning and thus seamlessly integrate sparse NNs and FL paradigms. Our framework, named *Federated Dynamic Sparse Training* (**FedDST**), leverages sparsity as the unifying tool to save both communication and local training costs.
- By using flexible aggregation methods, we deployed FedDST on top of FedAvg (McMahan et al. 2017) with no additional transmission overhead from clients. As a general design principle, our method is readily extendable to other FL frameworks such as FedProx (Li et al. 2018). Furthermore, the notion of dynamic sparsity is found to accommodate local heterogeneity, as well as create the bonus effect of in-time self-ensembling, that improve FL performance even over the dense baseline.
- Extensive experiments demonstrate that FedDST dramatically improves communication efficiency on difficult problems with pathologically non-iid data distributions. Even in these non-iid settings, FedDST provides a 3% accuracy improvement over FedAvgM (Hsu, Qi, and Brown 2019) on CIFAR-10, while requiring only half the upload bandwidth. We also provide extensive ablation studies showing robustness of FedDST to reasonable variations of its parameters. These results suggest sparse training as the future "go-to" option for FL.

Related Work

Federated Learning

In *federated learning* (Kairouz et al. 2019), a set of clients $j \in [N]$ collaborate to learn one or multiple models with the involvement of a central coordinating server. Each client has a small set of training data $\mathcal{D}_j$ for local training, but to preserve user privacy, clients do not share their local training data. In this work, we attempt to learn a global model θ, and aim to minimize the global loss

$$\min_{\theta} \sum_{j\in[N]} \sum_{z\in\mathcal{D}_j} \ell(\theta; z). \tag{1}$$

In the FedAvg (McMahan et al. 2017; Hsu, Qi, and Brown 2019) family of algorithms, training proceeds in *communication rounds*. To start each round i, the server selects a set of clients C_i and sends the current server model parameters θ^i to the clients. Each client $j \in C_i$ performs E epochs of training on the received model using its local training set to produce parameters θ^i_j, which it uploads to the server; in FedAvg, client-local training is performed via SGD. The server then updates the global model with a weighted average of the sampled clients' parameters. Reddi et al. (Reddi et al. 2021) argue that the updates produced by clients can be interpreted and used as pseudo-gradients. They thus generalize FedAvg into the FedOpt framework, which allows "plugging in" different client and server optimizers.

Real-world FL settings present many challenges due to non-iid data distributions, client heterogeneity, and limited compute, memory, and bandwidth at the edge (Kairouz et al. 2019; Hong et al. 2021). Heterogeneity in the compute capabilities of clients causes the so-called *straggler problem*, in which certain clients take "too long" to form model updates and the server must proceed without them. Hsu et al (Hsu, Qi, and Brown 2019), in FedAvgM, demonstrate that adding a momentum term to the client optimizer consistently improves performance in non-iid settings. Li et al. (Li et al. 2018) propose FedProx which adds a *proximal penalty* to FedAvg and allows stragglers to submit partial updates.

For communication-efficient FL, Konecny et al. (Konečný et al. 2017) distinguish between *sketched updates*, in which model updates are compressed during communication but not during local training; and *structured updates*, in which local training is performed directly on a compressed representation. Relatively little prior art discusses pruning or weight readjustment throughout the entire FL process.

Network Pruning

Model pruning aims to select a sparse subnetwork from a larger NN by removing connections. Traditionally, pruning methods start from a highly overparameterized trained model, remove connections, and fine-tune the pruned model. Common goals of pruning include saving compute, memory, communication, or other resources. Many selection criteria are possible, including weight magnitude (Han, Mao, and Dally 2015), optimal brain damage (LeCun, Denker, and Solla 1990; Hassibi et al. 1993; Dong, Chen, and Pan 2017), zero activations (Hu et al. 2016), and Taylor expansions (Molchanov et al. 2017).

Other recent studies have proposed a variety of algorithms to perform "single-shot" pruning at initialization. Lee et al. (Lee, Ajanthan, and Torr 2019) select connections at initialization by sampling a minibatch and sorting connections by their sensitivity. Wang et al. (Wang, Zhang, and Grosse 2020) similarly sample a minibatch at initialization but instead attempt to preserve gradient flow after pruning.

Dynamic Sparse Training

Dynamic sparse training (DST) shifts the selected subnetwork regularly throughout the training process, maintaining a constant number of parameters throughout. The seminal work (Mocanu et al. 2018) proposed the SET algorithm which iteratively prunes the smallest magnitude weights and grows random connections. SET also maintains a particular distribution of model density by layer, following the Erdős-Rényi random graph topology which scales the density of a layer with the number of input and output connections. In RigL (Evci et al. 2020), the authors initialize the sparsity mask randomly and perform layer-wise magnitude pruning and gradient-magnitude weight growth. As in (Mocanu et al. 2018), they follow particular layer-wise sparsity distributions and introduce the ERK sparsity distribution for convolutional layers, thus scaling their density by both number of connections and kernel size. Liu et al. (2021c) demonstrate the benefits that DST gains from parameter exploration; specifically, by exploring a number of possible sparse networks, DST is able to effectively perform "temporal self-ensembling," allowing for performance advantages even over dense networks (Liu et al. 2021a,b).

Pruning in Federated Learning

To our knowledge, there are only two works that address pruning throughout the FL process. PruneFL (Jiang et al. 2020) relies on an initial mask selected at a particular client, followed by a FedAvg-like algorithm that performs mask readjustment every ΔR rounds. Training is then performed via sparse matrix operations. On mask readjustment rounds, clients are required to upload full dense gradients which the server uses to form the aggregate gradient g. When selecting a mask, the indices j corresponding to prunable weights are sorted by g_j^2/t_j, where t_j is an estimate of the time cost of retaining connection j in the network. The estimates t_j are determined experimentally by measuring the time cost of one round of FL with various sparsities.

LotteryFL (Li et al. 2020) takes inspiration from LG-FedAvg (Liang et al. 2020) and allows clients to maintain local representations by selecting a local subset of the global network. It can also be described as an extension of FedAvg in which each client c maintains a separate mask m_c. At each round r, selected clients evaluate their subnetworks $\theta^r \odot m_c^r$ using local validation sets. If the validation accuracy exceeds a predefined threshold and the client's current sparsity $\|m_c^r\|_0$ is less than the target sparsity, then magnitude pruning is performed to produce a new mask m_c^{r+1} and the corresponding weights are reset to their initial values.

Comparing FedDST to prior pruning works in FL. First, unlike LotteryFL, which produces a system of sparse models that only perform well on local datasets, FedDST produces one global sparse model, dynamic over time, that performs well everywhere. FedDST performs mask readjustments on both clients and server but these are relatively low-overhead operations (layer-wise magnitude pruning and gradient-magnitude growth). Moreover, FedDST transmits neither dense models nor gradients, and does not train a dense model even at the very beginning: this is in sharp contrast to LotteryFL and makes FedDST significantly lighter.

Second, unlike PruneFL, FedDST is designed for a challenging, realistic non-iid FL setting. For this reason, FedDST's aggregation and dynamic sparse training, which allow mask readjustments at the client, provide resilience to non-iid data in a way that PruneFL's adaptive pruning criteria cannot achieve. In particular, PruneFL's clients do not readjust masks after the first round; they instead transmit gradients to the server on certain rounds, and the server uses gradient magnitudes and layer times to decide the mask for the next round. Since data heterogeneity means that gradient magnitudes cannot be directly compared between clients, gradient aggregation is inherently unstable. FedDST provides stable updates by deciding on the mask at the server using only weight magnitudes and mask "votes" submitted by the clients. Our experiments demonstrate advantage of FedDST vs. PruneFL in terms of the generalization power.

Algorithm 1: Overview of the proposed Federated Dynamic Sparse Training (FedDST).

```
Input: Clients $[N]$ with local datasets $\mathcal{D}_i$
Sparsities by layer $\mathbb{S} = \{s^1, \dots, s^L\}$
Update schedule $\Delta R, R_{end}, \alpha^r$
Initialize server model $(\theta^1, m^1)$ at sparsity
  $\|m^1\|_0 = S$;
for each round $r \in [R]$ do
    Sample clients $C_r \subset [N]$;
    Transmit the server model $(\theta^r, m^r)$ to all clients
      $c \in C_r$;
    for each client $c \in C_r$ do in parallel
        Receive $(\theta_c^r, m_c^r) \leftarrow (\theta^r, m^r)$ from the
          server;
        for each epoch $e \in [E]$ do
            Sample a minibatch $\mathcal{B}$ from $\mathcal{D}_c$;
            Perform one step of local training of local
              sparse network $\theta_c^r \odot m_c^r$ on $\mathcal{B}$;
            if $r \mod \Delta R = 0$ and $e = E_p$ and
              $r < R_{end}$ then
                Perform layer-wise magnitude
                  pruning
                  $(\theta_c^r, m_c^r) \leftarrow \text{prune}(\theta_c^r; \mathbb{S}_{1-\alpha^r})$ to
                  attain sparsity distribution $\mathbb{S}_{1-\alpha^r}$;
                Perform layer-wise gradient
                  magnitude growth
                  $(\theta_c^r, m_c^r) \leftarrow \text{grow}(\theta_c^r, g_c^r; \mathbb{S})$ to attain
                  sparsity distribution $\mathbb{S}$;
            end
        end
        Transmit the new $(\theta_c'^r, m_c'^r)$ to the server (do
          not transmit the mask if it has not changed);
    endfor
    Receive the updated client-local networks and
      masks $(\theta_c'^r, m_c'^r)$ from clients $c \in C_r$;
    Aggregate networks
      $\theta^{r+1/2} \leftarrow A(\{\theta_c^{r+1}, m_c^{r+1}\}_{c \in C_r})$;
    Perform layer-wise magnitude pruning
      $(\theta^{r+1}, m^{r+1}) \leftarrow \text{prune}(\theta^{r+1/2}; \mathbb{S})$ to attain
      sparsity distribution $\mathbb{S}$;
end
```

Thanks to the fixed sparsity budget throughout training, FedDST updates require very little network bandwidth, even in the worst case. Though PruneFL also transmits sparse updates for most of the rounds, it transmits full dense gradients to the server every few rounds to facilitate mask readjustments. LotteryFL requires clients to transmit dense models unless their accuracy meets a certain threshold, so dense transmissions happen even more frequently.

Methodology

FedDST provides a fully federated approach to dynamic sparse training of NNs. In this method, we aim to learn a single model that provides good accuracy to all clients, while also consuming minimal compute, memory, and communication resources. Our method is designed to perform well even in pathologically non-iid settings.

FedDST: Overview of the General Framework

We begin on the server by initializing a server network θ^1 and a sparse mask m^1, following the layer-wise sparsity distribution described in (Evci et al. 2020). At each round r, the server samples clients C_r. The server network and mask are sent to clients $c \in C_r$. Each client performs E epochs of local training. After the E_p-th epoch of local training, clients perform a *mask readjustment*, which reallocates α_r of the model mass to different connections. Readjustment is only performed on certain rounds, and the frequency of readjustment is specified by ΔR.

The selected clients upload their new sparse network and mask (if needed) to the server, and the server aggregates the received information to produce new global parameters and mask (θ^{r+1}, m^{r+1}). Server-side aggregation methods are discussed later in this section.

The client-side mask readjustment procedure is familiar and takes inspiration from RigL (Evci et al. 2020). The goal of the mask readjustment is to reallocate α_r of the model mass in order to seek a more effective subnetwork. We first prune the network to an even higher sparsity $S + (1 - S)\alpha_r$, while maintaining the same *distribution* of weights. Then, we regrow the same number of weights that were pruned, via gradients g_c^r, returning to the same original sparsity. Because different clients in a round may produce different masks, the server has to aggregate multiple sparse networks that may have explored the mask space in completely different directions. As in (Evci et al. 2020; Dettmers and Zettlemoyer 2019), we use a *cosine decay* update schedule for α_r,

$$\alpha_r = \frac{\alpha}{2}\left(1 + \cos\left(\frac{(r-1)\pi}{R_{end}}\right)\right). \tag{2}$$

On the first round, we have $\alpha_1 = \alpha$, and the proportion of weights reallocated each time decays to 0 at round R_{end}. The parameter α controls the tradeoff between exploration of the mask space and agreement between clients on mask decisions. Larger values of α encourage moving more quickly around the mask space, whereas smaller values encourage agreement and incremental adjustment of the mask. We explore the effect of α in the experiments.

Server Aggregation with Robustness to Heterogeneity Because the server does not directly receive gradients from clients, it must decide on the mask for the next round using only the parameters and masks received from the clients. From this, we define the *sparse weighted average*:

$$\theta^{r+1/2}[i] \leftarrow \frac{\sum_{c\in C_r} n_c \theta_c'^r[i] m_c'^r[i]}{\sum_{c\in C_r} n_c m_c'^r[i]}. \tag{3}$$

This method takes inspiration from the weighted averages used in FedAvg (McMahan et al. 2017), in which the effect of a particular client on a particular parameter is weighted by the dataset size at that client. However, the sparse weighted average also ignores parameter values from clients that did not provide any value. In particular, if a client has pruned out a weight, that client is ignored for the purposes of aggregation of that weight. Especially in pathologically non-iid FL settings, we find FedDST to benefit greatly from this mask aggregation method. Magnitudes of stochastic gradients cannot be directly compared between clients because data distribution varies greatly between clients. For this reason, methods such as PruneFL exhibit mask instability problems in the same setting. FedDST uses magnitudes of weights and "votes" from clients to solve this problem.

Accuracy Gains from Dynamic Sparsity: A Spatial-Temporal Ensembling Effect. Our experiments show that FedDST gains not only communication/computational savings, but also performance improvements: this is particularly notable in highly non i.i.d. settings. We attribute the "less is more" phenomenon to an underlying spatial-temporal "ensembling effect" allowed uniquely by FedDST.

On the "spatial" side, we refer to the fact that in FedDST, one mask is sent from the cloud to all clients, yet each client can re-adjust their mask and re-sample the weights according to its non i.i.d. local data. These new sparse masks and weights are periodically re-assembled in the cloud, reminiscent of the famous model sub-sampling and ensembling effect of "dropout" (Hinton et al. 2012), now along FL's spatial dimension (across clients). That is, each client can be viewed as a differently sampled subnetwork of the dense cloud model (not random, but "learned dropout"), and such can provide a regularization effect on training more robust weights for the cloud model by ensembling those subnetwork weights. Note that this similar effect does not take place in PruneFL, where all clients share one mask at each time. While LotteryFL also allows each client to have its own mask, it comes with heavy local computational overhead. On the "temporal" side, FedDST explores the mask space through time while also learning weights θ^r. Taken together, it allows for a continuous parameter exploration across training, taming a *space-time over-parameterization* (Liu et al. 2021c), which can significantly improve the expressibility and generalizability of sparse training.

Extending FedDST to Other FL Frameworks From the basic example above, FedDST can easily accommodate different local and global optimizers, as described in (Reddi et al. 2021). Algorithm 1 shows a general outline of FedDST, where local training can use any optimizer as needed. For example, we use SGD with momentum as described in (Hsu, Qi, and Brown 2019) as the local optimizer in our experiments. The pseudo-gradient generated by the aggregated sparse updates can also be used with other global optimizers.

FedDST is also compatible with FedProx (Li et al. 2018) and its proximal penalty $\|\theta_c^r - \theta^r\|_2$. However, if this penalty is directly used for mask readjustment, the penalty will act as a weight decay term on weights that have been pruned out. These weights will thus be less likely to be selected for regrowth. Therefore, in client growth, we use gradients corresponding to the loss without the proximal term.

Communication and Local Training Savings

Communication Analysis: FedDST provides significant competitive bandwidth savings at both uploading and downloading links. Let n denote the number of parameters in the network. In FedDST, the same sparsity S is maintained at each round and masks are only uploaded and downloaded a maximum of once every ΔR rounds, so FedDST has an average upload and download cost of $\left(32(1-S)+\frac{1}{\Delta R}\right)n$ bits per client per round before R_{end}, and a cost of $32(1-S)n$ after R_{end}. Furthermore, the maximum upload or download cost at any client is $(32(1-S)+1)\,n$ bits, so FedDST successfully avoids placing the burden of large uploads on any one client for any round. We believe that these upload cost savings should also help significantly to combat the straggler problem (Li et al. 2018; Kairouz et al. 2019) in practice, as clients with much slower connections will never be forced to upload full models or gradients. Furthermore, in commercial FL systems with a large number of clients, it is likely that a particular client will only be selected once. With this upper limit on bandwidth costs, FedDST also makes it practical to learn models via FL, even on cellular networks.

This is in contrast to PruneFL, which requires clients to send full gradients to the server every ΔR rounds, leading to an average upload cost of $\left(32(1-S)+\frac{32}{\Delta R}\right)n$ bits per client per round, and a maximum upload cost of $(32(1-S)+32)\,n$. Thus FedDST is no worse than PruneFL during normal rounds, and at sparsity $S = 0.8$, FedDST is $5\times$ cheaper than PruneFL per round with respect to upload cost during mask readjustment rounds. Finally, PruneFL relies on a single client to provide an initial sparsity pattern. Because of local client heterogeneity in FL, the mask produced by one client tends to reflect the local data distribution of that client. For the same reason, later updates to the sparsity pattern exhibit instability. FedDST completely bypasses these problems by starting with a random mask followed by the sparse weighted average aggregation.

Computation Analysis: FedDST also saves considerable local computational workloads in FL by maintaining sparse networks throughout the FL process. No part of FedDST requires dense training. In terms of FLOP savings, this allows us to skip most of the FLOPs in both training and inference, proportional to the sparsity of the model. For example, at 80% sparsity, for forward computation on the network we use on CIFAR-10, only 0.8 MFLOPs are required, while 4.6 MFLOPs are required for the dense network. For the same reason, the experiments against cumulative upload data caps also roughly reflect the accuracy at different FLOP limits.

Note that following the convention of (Mocanu et al. 2018; Evci et al. 2020; Liu et al. 2021c), FedDST so far only considers element-wise unstructured sparsity. Unstructured sparsity was traditionally considered less translatable into real hardware benefits due to irregular access (Wen et al. 2016). However, at 70%-90% unstructured sparsity, XNNPack (Elsen et al. 2020) recently showed significant speedups over dense baselines on smartphones, motivating our future work to optimize practical local training speedups on a FL hardware platform, and to incorporate structured sparsity in dynamic sparse training (You et al. 2020).

	Best accuracy encountered at cumulative upload capacity [GiB]			
Method	1	2	3	4
FedAvgM	85.25	96.32	97.16	97.53
FedProx ($\mu = 1$)	82.34	95.84	97.16	97.54
FedAvgM bfloat16	77.41	90.13	96.88	97.46
RandomMask	93.61	96.89	97.5	97.72
GraSP	61.95	86.06	94.15	96.29
PruneFL	78.12	89.29	91.65	93.26
FedDST	**96.10**	**97.35**	**97.67**	**97.83**
FedDST+FedProx	95.35	96.97	97.26	97.81

Table 1: Accuracy of FedDST and other methods given cumulative upload bandwidth limits, on non-iid MNIST. We fix $S = 0.8$ for sparse methods, $\alpha = 0.05$ for DST methods, and $\mu = 1$ for the proximal penalty.

	Best accuracy encountered at cumulative upload capacity [GiB]			
Method	4	8	12	16
FedAvgM	24.43	33.87	37.07	40.52
FedProx ($\mu = 1$)	23.54	34.01	39.08	42.56
FedAvgM bfloat16	22.58	34.05	37.10	41.65
RandomMask	33.98	41.86	45.99	48.01
GraSP	15.68	29.5	39.7	44.85
PruneFL	17.37	25.3	30.88	35.29
FedDST	**35.41**	42.27	**46.72**	**50.67**
FedDST+FedProx	33.03	**43.18**	46.66	49.69

Table 2: Accuracy of FedDST and other methods given cumulative upload bandwidth limits, on non-iid CIFAR-10. We fix $S = 0.8$ for sparse methods, $\alpha = 0.001$ for FedDST alone, $\alpha = 0.01$ for FedDST with the proximal penalty, and $\mu = 1$ for the proximal penalty.

	Best accuracy encountered at cumulative upload capacity [GiB]			
Method	8	16	24	32
FedAvgM	6.66	9.29	10.13	10.94
FedProx ($\mu = 1$)	2.74	3.87	4.42	5.12
FedAvgM bfloat16	7.78	9.92	10.92	12.02
RandomMask	7.15	8.65	9.41	9.69
GraSP	4.45	6.61	7.78	8.37
PruneFL	5.78	8.10	9.44	10.02
FedDST	9.14	11.30	13.18	13.96
FedDST+FedProx	**9.40**	**11.29**	**13.46**	**14.57**

Table 3: Accuracy of FedDST and other methods given cumulative upload bandwidth limits, on non-iid CIFAR-100. We fix $S = 0.5$ for sparse methods, $\alpha = 0.01$ for FedDST, and $\mu = 1$ for the proximal penalty.

Experiments

We use MNIST (LeCun, Cortes, and Burges 2010) and CIFAR-10 (Krizhevsky 2009) datasets distributed among clients in a "pathologically non-iid" setting, similar to (McMahan et al. 2017) and matching the datasets used in (Li et al. 2020). We assume a total pool of 400 clients. Each client is assigned 2 classes and given 20 training images from each class. To distribute CIFAR-100 (Krizhevsky 2009) in a non-iid fashion, we use a $\mathrm{Dirichlet}(0.1)$ distribution for each class to distribute its samples among 400 clients, as in (Wang et al. 2020; Li, He, and Song 2021; Wang et al. 2021). These distributions represent a challenging and realistic training environment, with only a fraction of the training data available and distributed in a severely non-iid fashion among clients. We provide more details for all datasets in the appendix.

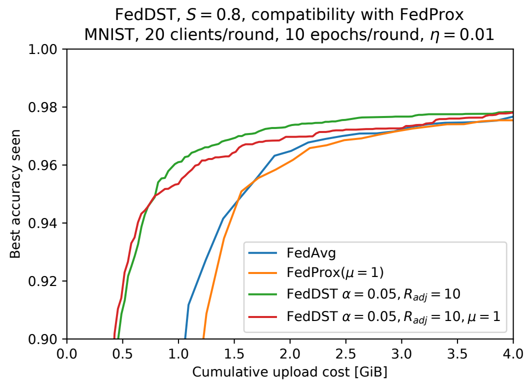
Main Results

We compare FedDST to competitive state-of-the-art baselines. Both FedAvgM (Hsu, Qi, and Brown 2019) and FedProx (Li et al. 2018) are designed for non-iid settings. We include PruneFL (Jiang et al. 2020) as the other method involving dynamic sparsity in FL. As mentioned before, PruneFL relies on stochastic gradients uploaded by clients, so its masks exhibit instability between reconfigurations. In the cases where PruneFL converged, it selected an all-ones mask, recovering plain FedAvg. Note that because FedDST aims to produce a single model in each round that performs well on all clients, we do not compare to algorithms producing separate models for each client, such as LG-FedAvg (Liang et al. 2020) and LotteryFL (Li et al. 2020).

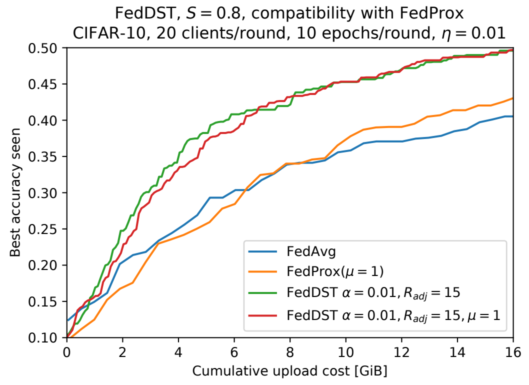
For RandomMask, we randomly sample weights at the server, then perform layer-wise magnitude pruning, following the ERK sparsity distribution (Evci et al. 2020), before the first round, and perform FedAvgM on this sparse network. The random mask selected is held constant as the global and local sparsity mask throughout training. RandomMask represents a strong communication-efficient baseline for this non-iid setting, in which the noisy magnitudes of stochastic gradients cause methods that rely on gradient aggregation, such as PruneFL, to fail. Furthermore, the data heterogeneity in this environment causes GraSP to select a mask that works well for the initially chosen client but does not work well as a global mask. Despite this challenging environment, FedDST still produces significant accuracy improvements over the strong RandomMask baseline.

Tables 1, 2, and 3 provide accuracies achieved by FedDST and other algorithms given certain upload bandwidth limits. We report the best test accuracy seen before each limit, averaged across 10 runs. FedDST consistently provides the best performance at any upload limit, halving upload cost with respect to FedAvgM in all settings we tested.

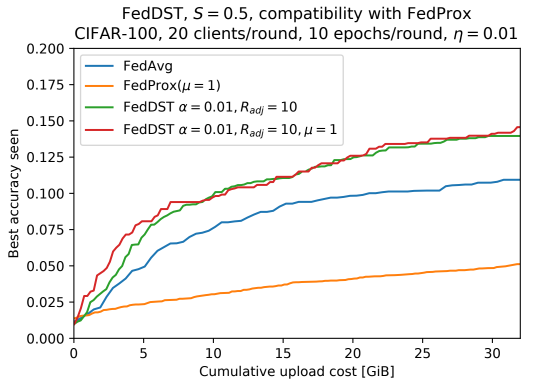
Compatibility with FedProx Our experimental results also confirm that FedDST is compatible with other FL frameworks, including FedProx. For this experiment, we add FedProx's proximal term and adjust FedDST's growth criterion as described in the overview. Figure 1 shows that applying FedDST to FedProx consistently improves its performance across datasets. FedDST is therefore a general framework that can be applied to various FL optimizers.



(a) FedDST+FedProx on non-iid MNIST



(b) FedDST+FedProx on non-iid CIFAR-10

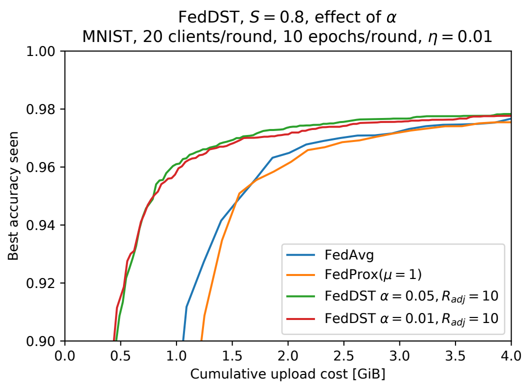


(c) FedDST+FedProx on non-iid CIFAR-100

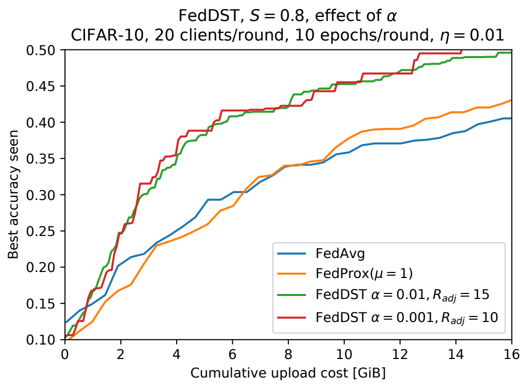
Figure 1: FedDST is compatible with other popular FL frameworks, such as FedProx. This shows that FedDST is a general framework that can also apply to other optimizers.

Insensitivity to Variations in α

In Figure 2, we show that FedDST performs well with any reasonable value of α, even in the non-iid setting we explore in this paper. The non-iid distribution targeted by FedDST leads to noisy stochastic gradients produced by clients. Thus, directly using the magnitudes of gradients to produce mask readjustments causes large variations in the mask. FedDST effectively sidesteps this problem by only using the top local stochastic gradients to "vote" for particular weight in-



(a) FedDST on non-iid MNIST



(b) FedDST on non-iid CIFAR-10

Figure 2: FedDST is insensitive to certain variations in α.

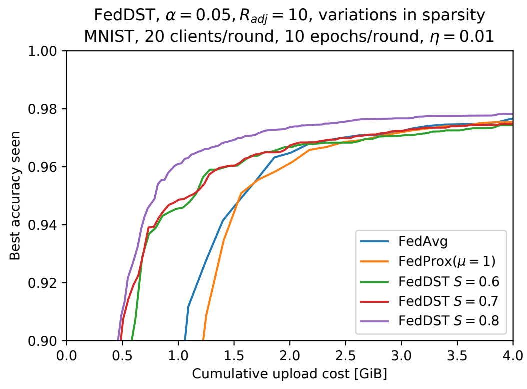
dices. The parameter α specifies how many of these "votes" should be submitted to the server. However, suitable values for α are still smaller than in RigL; for $\alpha \in [0.001, 0.05]$, FedDST significantly outperforms other methods.

Performance at Different Sparsity Levels

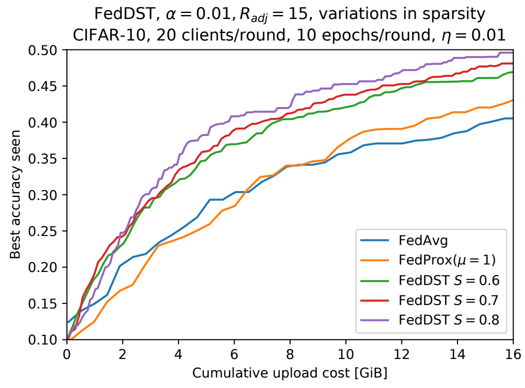
In Figure 3, we show that FedDST's performance is robust to variations in sparsity. As sparsity directly leads to communication savings, FedDST performs best at relatively high sparsities, even on the lightweight NNs we test here. However, even at untuned sparsity levels, FedDST performs reasonably well and converges much more quickly than FedAvgM, especially at the beginning of the FL process. Hence we run all FedDST experiments at sparsity 0.5 or 0.8.

Conclusion and Broader Impacts

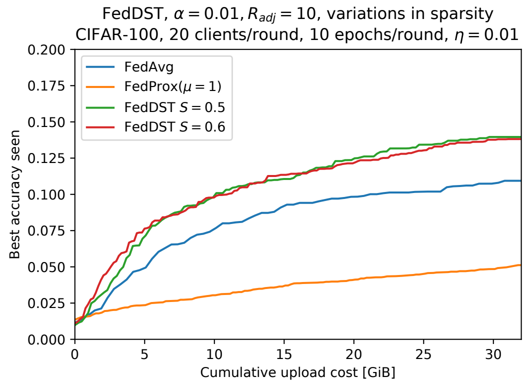
We introduce *Federated Dynamic Sparse Training* (FedDST), a powerful framework for communication-efficient federated learning via dynamic sparse training, which works well even on pathologically non-iid datasets. We show experimentally that FedDST consistently outperforms competing algorithms, producing up to 3% better accuracy than FedAvgM with half the upload bandwidth on non-iid CIFAR-10. We further demonstrate that FedDST is compatible with other popular federated optimization frameworks such as



(a) FedDST with varying sparsity on non-iid MNIST



(b) FedDST with varying sparsity on non-iid CIFAR-10



(c) FedDST with varying sparsity on non-iid CIFAR-100

Figure 3: FedDST is robust to sparsity variations, and performs the best at reasonable sparsities.

FedProx. Our results indicate a bright future for sparsity in even the most difficult FL settings.

FL has the potential to enable learning NNs in situations in which privacy of user data is of paramount importance. For example, medical datasets are bound by strong legal restrictions and cannot be exchanged between clients. Furthermore, FL can provide stronger privacy guarantees in existing machine learning settings involving user data. With its focus on improving communication efficiency and performance, FedDST makes more FL settings practical in the wild.

Acknowledgements

The authors would like to recognize that the corresponding author, Xiaohan Chen, was the other student author driving the idea and experimental design in this paper.

Portions of this research (by S. Bibikar and H. Vikalo) were sponsored by the Army Research Office and were accomplished under Cooperative Agreement Number W911NF-19-2-0333. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

X. Chen and Z. Wang's research were supported in part by the NSF Real-Time Machine Learning program (Award Number: 2053279), and the NSF AI Institute for Foundations of Machine Learning (IFML).

References

Chen, T.; Frankle, J.; Chang, S.; Liu, S.; Zhang, Y.; Carbin, M.; and Wang, Z. 2021. 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 (NeurIPS)*.

Dettmers, T.; and Zettlemoyer, L. 2019. Sparse Networks from Scratch: Faster Training without Losing Performance. arXiv:1907.04840.

Dong, X.; Chen, S.; and Pan, S. 2017. Learning to Prune Deep Neural Networks via Layer-wise Optimal Brain Surgeon. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Elsen, E.; Dukhan, M.; Gale, T.; and Simonyan, K. 2020. Fast sparse convnets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 14629–14638.

Evci, U.; Gale, T.; Menick, J.; Castro, P. S.; and Elsen, E. 2020. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, 2943–2952. PMLR.

Frankle, J.; and Carbin, M. 2018. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*.

Frankle, J.; Dziugaite, G. K.; Roy, D.; and Carbin, M. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, 3259–3269. PMLR.

Gale, T.; Elsen, E.; and Hooker, S. 2019. The state of sparsity in deep neural networks. 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. In *International Conference on Learning Representations*.

Hassibi, B.; Stork, D. G.; Wolff, G.; and Watanabe, T. 1993. Optimal Brain Surgeon: Extensions and Performance Comparisons. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, 263–270. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580.

Hong, J.; Zhu, Z.; Yu, S.; Wang, Z.; Dodge, H. H.; and Zhou, J. 2021. Federated Adversarial Debiasing for Fair and Transferable Representations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 617–627.

Hsu, T.-M. H.; Qi, H.; and Brown, M. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. arXiv:1909.06335.

Hu, H.; Peng, R.; Tai, Y.-W.; and Tang, C.-K. 2016. Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures. arXiv:1607.03250.

Jiang, Y.; Wang, S.; Valls, V.; Ko, B. J.; Lee, W.-H.; Leung, K. K.; and Tassiulas, L. 2020. Model Pruning Enables Efficient Federated Learning on Edge Devices. arXiv:1909.12326.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; D'Oliveira, R. G. L.; Rouayheb, S. E.; Evans, D.; Gardner, J.; Garrett, Z.; Gascón, A.; Ghazi, B.; Gibbons, P. B.; Gruteser, M.; Harchaoui, Z.; He, C.; He, L.; Huo, Z.; Hutchinson, B.; Hsu, J.; Jaggi, M.; Javidi, T.; Joshi, G.; Khodak, M.; Konečný, J.; Korolova, A.; Koushanfar, F.; Koyejo, S.; Lepoint, T.; Liu, Y.; Mittal, P.; Mohri, M.; Nock, R.; Özgür, A.; Pagh, R.; Raykova, M.; Qi, H.; Ramage, D.; Raskar, R.; Song, D.; Song, W.; Stich, S. U.; Sun, Z.; Suresh, A. T.; Tramèr, F.; Vepakomma, P.; Wang, J.; Xiong, L.; Xu, Z.; Yang, Q.; Yu, F. X.; Yu, H.; and Zhao, S. 2019. Advances and Open Problems in Federated Learning. arXiv:1912.04977.

Konečný, J.; McMahan, H. B.; Yu, F. X.; Richtárik, P.; Suresh, A. T.; and Bacon, D. 2017. Federated Learning: Strategies for Improving Communication Efficiency. arXiv:1610.05492.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

LeCun, Y.; Cortes, C.; and Burges, C. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

LeCun, Y.; Denker, J.; and Solla, S. 1990. Optimal Brain Damage. In Touretzky, D., ed., *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Lee, N.; Ajanthan, T.; and Torr, P. 2019. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*.

Li, A.; Sun, J.; Wang, B.; Duan, L.; Li, S.; Chen, Y.; and Li, H. 2020. LotteryFL: Personalized and Communication-Efficient Federated Learning with Lottery Ticket Hypothesis on Non-IID Datasets. arXiv:2008.03371.

Li, Q.; He, B.; and Song, D. 2021. Model-Contrastive Federated Learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10708–10717. Los Alamitos, CA, USA: IEEE Computer Society.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2018. Federated optimization in heterogeneous networks. arXiv:1812.06127.

Liang, P. P.; Liu, T.; Ziyin, L.; Salakhutdinov, R.; and Morency, L.-P. 2020. Think locally, act globally: Federated learning with local and global representations. arXiv:2001.01523.

Liu, S.; Chen, T.; Atashgahi, Z.; Chen, X.; Sokar, G.; Mocanu, E.; Pechenizkiy, M.; Wang, Z.; and Mocanu, D. C. 2021a. Deep Ensembling with No Overhead for either Training or Testing: The All-Round Blessings of Dynamic Sparsity. arXiv:2106.14568.

Liu, S.; Chen, T.; Chen, X.; Atashgahi, Z.; Yin, L.; Kou, H.; Shen, L.; Pechenizkiy, M.; Wang, Z.; and Mocanu, D. C. 2021b. Sparse Training via Boosting Pruning Plasticity with Neuroregeneration. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Liu, S.; Yin, L.; Mocanu, D. C.; and Pechenizkiy, M. 2021c. Do We Actually Need Dense Over-Parameterization? In-Time Over-Parameterization in Sparse Training. arXiv:2102.02887.

Ma, X.; Yuan, G.; Shen, X.; Chen, T.; Chen, X.; Chen, X.; Liu, N.; Qin, M.; Liu, S.; Wang, Z.; et al. 2021. Sanity Checks for Lottery Tickets: Does Your Winning Ticket Really Win the Jackpot? *Advances in Neural Information Processing Systems*, 34.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A.; and Zhu, J., eds., *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, 1273–1282. Fort Lauderdale, FL, USA: PMLR.

Mocanu, D. C.; Mocanu, E.; Stone, P.; Nguyen, P. H.; Gibescu, M.; and Liotta, A. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1): 2383.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2017. Pruning Convolutional Neural Networks for Resource Efficient Inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Reddi, S. J.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečný, J.; Kumar, S.; and McMahan, H. B. 2021. Adaptive Federated Optimization. In *International Conference on Learning Representations*.

Ribero, M.; and Vikalo, H. 2020. Communication-efficient federated learning via optimal client sampling. arXiv:2007.15197.

Wang, C.; Zhang, G.; and Grosse, R. 2020. Picking Winning Tickets Before Training by Preserving Gradient Flow. In *International Conference on Learning Representations*.

Wang, H.; Yurochkin, M.; Sun, Y.; Papailiopoulos, D.; and Khazaeni, Y. 2020. Federated Learning with Matched Averaging. In *International Conference on Learning Representations*.

Wang, L.; Xu, S.; Wang, X.; and Zhu, Q. 2021. Addressing Class Imbalance in Federated Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11): 10165–10173.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2082–2090.

You, H.; Li, C.; Xu, P.; Fu, Y.; Wang, Y.; Chen, X.; Baraniuk, R. G.; Wang, Z.; and Lin, Y. 2020. Drawing Early-Bird Tickets: Toward More Efficient Training of Deep Networks. In *Int. Conf. on Learning Representations*.