Time-Correlated Sparsification for Communication-Efficient Federated Learning

Emre Ozfatura[†], Kerem Ozfatura[‡] and Deniz Gündüz [†]
[†]Information Processing and Communications Lab, Dept. of Electrical and Electronic Engineering, Imperial College London
[‡]Department of Computer Science, Ozyegin University
{m.ozfatura,d.gunduz}@imperial.ac.uk,kerem.ozfatura@ozu.edu.tr

***Abstract*—Federated learning (FL) enables multiple clients to collaboratively train a shared model,with the help of a parameter server (PS), without disclosing their local datasets. However, due to the increasing size of the trained models, the communication load due to the iterative exchanges between the clients and the PS often becomes a bottleneck in the performance. Sparse communication is often employed to reduce the communication load, where only a small subset of the model updates are communicated from the clients to the PS. In this paper, we introduce a novel time-correlated sparsification (TCS) scheme, which builds upon the notion that sparse communication framework can be considered as identifying the most significant elements of the underlying model. Hence, TCS exploits the correlation between the sparse representations at consecutive iterations in FL, so that the overhead due to encoding of the sparse representation can be significantly reduced without compromising the test accuracy. Through extensive simulations on the CIFAR-10 dataset, we show that TCS can achieve centralized training accuracy with** 100 **times sparsification, and up to** 2000 **times reduction in the communication load when employed with quantization**

I. Introduction

The key dilemma in the employment of deep neural networks (DNNs) is that often the training data is distributed across multiple institutions, and cannot be aggregated for centralized training due to the sensitivity of data [1]. On the other hand, data available at a single institution, such as a single bank, hospital, or a factory, may not be sufficient to train a "sufficiently good" model with the desired generalization capabilities. *Federated learning (FL)* framework has been introduced to address this dilemma [2] by orchestrating the participating clients with the help of a central, so-called parameter server (PS), such that they can perform local training using their datasets first, and seek consensus on the global model by communicating with each other. Since it requires the exchange of a large number of parameter values periodically, the communication load can be a major bottleneck, particularly when the underlying model is of high complexity. Besides, in FL, communication often takes place over bandwidth-limited channels [3], [4], which further increases the latency. To this end, communication-efficient designs are one of the key requirements for the successful implementation of collaborative training over multiple clients in a federated manner.

This work was funded by the European Research Council (ERC) through Starting Grant BEACON (no. 677854) and by the UK EPSRC (grant no. EP/T023600/1).

Algorithm 1 Federated Averaging (FedAvg)

```
1: for $t = 1, 2, \dots$ do
2:     for $n = 1, \dots, N$ do in parallel
3:         Pull $\boldsymbol{\theta}_t$ from PS: $\boldsymbol{\theta}^0_{n,t} = \boldsymbol{\theta}_t$
4:         for $\tau = 1, \dots, H$ do
5:             Compute SGD: $\mathbf{g}^\tau_{n,t} = \nabla_{\boldsymbol{\theta}} f_n(\boldsymbol{\theta}^{\tau-1}_{n,t}, \zeta_{n,\tau})$
6:             Update model: $\boldsymbol{\theta}^\tau_{n,t} = \boldsymbol{\theta}^{\tau-1}_{n,t} - \eta_t g^\tau_{n,t}$
7:     Federated Averaging: $\boldsymbol{\theta}_{t+1} = \frac{1}{|\mathcal{S}_t|}\sum_{n\in\mathcal{S}_t} \boldsymbol{\theta}^H_{n,t}$
```

A. Preliminaries

The objective of FL is to solve the following optimization problem over N clients

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N} \underbrace{\mathbb{E}_{\zeta_n\sim\mathcal{D}_n} f(\boldsymbol{\theta}, \zeta_n)}_{:=f_n(\boldsymbol{\theta})}, \tag{1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ denotes the model parameters, ζ_n is a random data sample, $\mathcal{D}_n$ denotes the dataset of client n, and f is the problem specific empirical loss function. At each iteration of FL, each client aims to minimize its local loss function $f_n(\boldsymbol{\theta})$ using the *stochastic gradient descent* method. Then, the clients seek a consensus on the model with the help of the PS. The most widely used consensus strategy is to periodically average the locally optimized model parameters, which is referred to as *federated averaging (FedAvg)* [2]. The FedAvg procedure is summarized in Algorithm 1.

At the beginning of iteration t, each client pulls the current global parameter vector $\boldsymbol{\theta}_t$ from the PS. In order to reduce the communication load, each client performs H local updates before the consensus step, as illustrated in Algorithm 1 (lines 5-6), where $\mathbf{g}^\tau_{n,t} = \nabla_{\boldsymbol{\theta}} f_n(\boldsymbol{\theta}^{\tau-1}_{n,t}, \zeta_{n,\tau})$ is the gradient estimate of the n-th client at τ-th local iteration based on the randomly sampled local data $\zeta_{n,\tau}$, and η_t is the learning rate.

We note that when $H = 1$, clients can send their local gradient estimates instead of updated models, and this particular implementation is called federated SGD (FedSGD) [2]. For the sake of completeness, we also want to highlight that when the number of participating clients are large, e.g., FL across mobile devices, the PS can choose a subset of the clients for the consensus to reduce the communication overhead.

However, in the scope of this work, we consider a scenario with a moderate number of clients, all of which participate in all the iterations of the learning process. This would be the case when the clients represent institutions, e.g., hospitals or banks; and hence, client selection is not required. In addition to multiple local iterations, we can also employ compression of model updates in order to reduce the communication load from the clients to the PS at each iteration [5]. Next, we briefly explain common compression strategies used in conveying the model updates from the clients to the PS in an efficient manner.

B. Compressed Communication

The global model update in Algorithm 1 (line 8) can be equivalently written in the following form:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{1}{N}\sum_{n=1}^{N} \underbrace{\sum_{\tau=1}^{H} -\eta_t \mathbf{g}_{n,t}^{\tau}}_{\Delta\boldsymbol{\theta}_{n,t}}, \tag{2}$$

where we call the term $\Delta\boldsymbol{\theta}_{n,t}$ the model difference of the nth client at iteration t. Hence, each client can send the model difference instead of the updated model, and the compression is applied to this model difference. *Sparsification* and *quantization* are two frequently used approaches for communication efficient FL.

Quantization aims to represent each element of $\Delta\boldsymbol{\theta}_{n,t}$ with fewer bits to reduce the communication load [6]–[9], where, initially, each element represented according to floating point precision with 32 bits. Hence, with quantization it is possible to achieve up to $\times 32$ reduction in the communication load [6], [8], [9].

Sparsification [10]–[13], instead, transforms a d dimensional vector into its sparse representation by mapping each element of vector to either its original value or zero, such that in the sparse representation only $\phi \cdot d$ number of elements have non-zero values, where ϕ denotes the sparsification ratio. Sparsification in the FL setup, according to the model update in (2), can be considered as applying a d-dimensional mask vector $\mathbf{m} \in \{0,1\}^d$ on $\Delta\boldsymbol{\theta}_{n,t}$; that is, $\hat{\Delta\boldsymbol{\theta}}_{n,t} = \mathbf{m} \otimes \Delta\boldsymbol{\theta}_{n,t}$, where $\otimes$ denotes element-wise multiplication.

For collaborative/distributed learning, sparsification is commonly adopted in practice, and we identify its two popular variations in the literature: *top-K sparsification* and *rand-K sparsification* [14]. Each variation has certain advantages and disadvantages; to summarize in a very broad sense, in general, top-K sparsification has a lower compression error compared to rand-K sparsification, especially for lower ϕ values; however, it is less efficient from the communication and processing perspectives. In rand-K sparsification, all the clients generate a random mask using the same seed; therefore, the information of non-zero positions is not required at the PS, which is not the case for top-K sparsification; that is, in addition to the sparse values, clients need to communicate the locations of these non-zero values to the PS. More detailed discussion on the comparison of these two sparsification strategies can be found in [15].

C. Sparse Network Architectures: Connections to Dynamic Network Pruning

Network pruning aims to reduce the size and complexity of DNNs [16], [17]. Given an initial DNN model $\boldsymbol{\theta}$, the objective of network pruning is to find a sparse version of $\boldsymbol{\theta}$, denoted by $\widetilde{\boldsymbol{\theta}}$, where only a small subset of the parameters are utilized with minimal loss in accuracy. In other words, the objective is to construct a d-dimensional mask vector $\mathbf{m}_p \in \{0,1\}^d$ to recover $\widetilde{\boldsymbol{\theta}} = \mathbf{m}_p \otimes \boldsymbol{\theta}$, where $||\mathbf{m}_p||_1 << d$. The 'lottery ticket hypothesis' [17] further states that such a mask can be employed throughout training to obtain the same level of accuracy with similar training time. We note that the existence of a "good" mask with $||\mathbf{m}_p||_1 << d$ and a similar test accuracy as the unpruned network implies the existence of a sparse communication strategy during training. If the optimal pruning mask $\mathbf{m}_p$ is used throughout training, this would also result in a significant reduction in the communication load by more than a factor of $\phi = ||\mathbf{m}_p||_1/d$, as the clients need to convey only $||\mathbf{m}_p||_1$ values, and there is no need to specify their locations. However, the optimal pruning mask typically cannot be determined at the beginning of the training process. Alternatively, in *dynamic network pruning* [16], an evolving sequence of masks are employed over iterations, where the pruning mask employed at each iteration is updated gradually.

D. Motivation and Contributions

At each iteration of the top-K sparsification strategy, each client constructs a mask vector $\mathbf{m}_{n,t}$ from scratch, independently of the previous iterations, although the gradient values, and hence, the top-K positions, exhibit certain correlation over time. The core idea of our work is to exploit this correlation to reduce the communication load resulting from the transmission of non-zero parameter locations. In other words, if the *significant* locations, those often observe large gradient values, were known, the clients could simply communicate the values corresponding to these locations without searching for the top-K locations, or specifying their indices to the PS, reducing significantly both the communication load and the complexity.

More specifically, inspired by the dynamic pruning technique [16], we propose a novel sparse communication strategy, called *time correlated sparsification (TCS)*, where we search for a "good" global mask $\mathbf{m}_t$ to be used by all the clients at iteration t, which evolves gradually over iterations. Client n uses a slightly personalized mask $\mathbf{m}_{n,t}$ based on the global mask $\mathbf{m}_t$, where

$$||\mathbf{m}_{n,t} - \mathbf{m}_t||_1 = \epsilon_t << \phi d. \tag{3}$$

The proposed TCS strategy exploits the correlation between $\mathbf{m}_{n,t}$ and $\mathbf{m}_t$ to reduce the communication load, similarly to data compression with side information [18], since $\mathbf{m}_t$ is already known to the PS. Therefore, only the locations of the small number of additional entries of $\mathbf{m}_{n,t}$ need to be conveyed to the PS. The presence of small variations between $\mathbf{m}_{n,t}$ and $\mathbf{m}_t$ serve two purposes: First, they are used to 'explore' a small portion of new locations to improve the current mask. Second, certain locations may become significant *temporarily* due to the accumulation of errors when an error feedback mechanism is employed [10], [19]. The advantages of the proposed TCS strategy can be summarized as follows:

- Compared to top-K sparsification, each client encodes and sends only a small number of non-sparse positions at each iteration, which reduces the number of transmitted bits. Furthermore, it makes high compression rates possible when TCS is combined with quantization.
- Although both TCS and top-K offers the same sparsification level in the uplink direction, TCS achieves much higher sparsification level in the downlink direction thanks to the correlation of the masks across clients.
- Finally, since the individual masks mostly coincide, in a federated edge learning (FEEL) scenario, where the clients communicate with the PS over a shared wireless channel, TCS allows employing over-the-air computation in an efficient manner and can take advantage of the superposition property of the wireless medium [20]–[24].

In Section III, through extensive simulations on the CIFAR-10 dataset, we show that TCS can achieve centralized training accuracy with 100 times sparsification, and up to 2000 times reduction in the communication load when employed together with quantization.

II. Time Correlated Sparsification (TCS)

A. Design principle

The main design principle behind TCS is employing two distinct mask vectors for sparsification: $\mathbf{m}_{global}$ is used to exploit previously identified important DNN parameters, whereas $\mathbf{m}_{local}$ explores new parameters. In particular, $\mathbf{m}_{global}$ is constructed based on the previous model update $\Delta\boldsymbol{\theta}$, motivated by the assumption that the most important DNN weights do not change significantly over iterations. Since all the clients receive the same global model update from the PS, mask vector $\mathbf{m}_{global}$ is identical for all the clients.

Let ϕ_{global} be the sparsification ratio for the mask $\mathbf{m}_{global}$ such that

$$||\mathbf{m}_{global}||_1 = K_{global} = \phi_{global} \cdot d \ . \tag{4}$$

At the beginning of iteration t, each client receives the global model difference $\Delta\boldsymbol{\theta}_{t-1}$ from the PS, and accordingly, obtains $\mathbf{m}_{global}$ by simply identifying the top K_{global} values in $|\Delta\boldsymbol{\theta}_{t-1}|$. In parallel, each client uses the received global model difference to recover $\boldsymbol{\theta}_t$.

Following the model update, each client $n \in [N]$ carries out the local SGD steps, and computes the local model difference $\Delta\boldsymbol{\theta}_{n,t}$. Finally, it obtains the sparse version of the local model difference, $\hat{\Delta}\boldsymbol{\theta}_{n,t}$, using the mask vector $\mathbf{m}_{global}$, i.e.,

$$\hat{\Delta}\boldsymbol{\theta}_{n,t} = \mathbf{m}_{global} \otimes \Delta\boldsymbol{\theta}_{n,t}. \tag{5}$$

We want to emphasize that since the PS sent out $\Delta\boldsymbol{\theta}_{t-1}$, mask vector $\mathbf{m}_{global}$ is known by the PS as well. Therefore, for each client it is sufficient to send only the non-zero values of $\hat{\Delta}\boldsymbol{\theta}_{n,t}$, without specifying their positions. Therefore, compared to the conventional top-K sparsification framework, TCS further reduces the communication load by removing the need to communicate the positions of the non-zero values of $\hat{\Delta}\boldsymbol{\theta}_{n,t}$.

However, the main drawback of the above approach is that, if $\mathbf{m}_{global}$ is used throughout the training process, the same subset of weights will be used for model update at all iterations. Therefore, the proposed sparsification strategy requires a feedback mechanism in order to explore new weights at each iteration to check whether there are more important weights to consider for model update.

To introduce such a feedback mechanism, each client n employs a second mask $\mathbf{m}^n_{local}$, which is unique to that client. The feedback mechanism works in the following way: given $\hat{\Delta}\boldsymbol{\theta}_{n,t}$, $\mathbf{m}^n_{local}$ is obtained as the vector of the greatest $K_{local} = \phi_{local} \cdot d$ entries of $|\Delta\boldsymbol{\theta}_{n,t} - \hat{\Delta}\boldsymbol{\theta}_{n,t}|$. Hence, at each iteration t, client n sends, $n \in [N]$,

$$\widetilde{\Delta}\boldsymbol{\theta}_{n,t} = \Delta\boldsymbol{\theta}_{n,t} \otimes (\mathbf{m}_{global} + \mathbf{m}^n_{local}) \tag{6}$$

to the PS.

Since the main purpose of $\mathbf{m}^n_{local}$ is to explore new important parameters, we assume that $\phi_{local} << \phi_{global}$. Assume that q bits are used to represent the value of each parameter. Then, using the global sparsification mask, the total number of bits to be conveyed to the PS is $K_{global} \cdot q$. For the feedback mechanism, in addition to q bits to represent each of the parameter values, $\log_2 d$ bits are required to inform the PS about each position of the parameter within the d-dimensional update vector. Hence, the total number of bits transmitted at each iteration is given by:

$$Q_{TCS} = q \cdot d \cdot (\phi_{local} + \phi_{global}) + \log_2 d \cdot d \cdot \phi_{local}. \tag{7}$$

Here, we would like to note that by utilizing a more efficient encoding strategy, it is possible to represent each position with $\log_2(1/\phi_{local}) + 2$ bits, instead of $\log_2 d$, which is more communication efficient when d is large. We refer the reader to [15] for the details of this encoding strategy. We want to emphasize that since $\phi_{local} << \phi_{global}$, the proposed TCS strategy is more communication efficient than top-K sparsification with the same ϕ_{global} value, such that $K = \phi_{global} \times d$. The total number of required bits for top-K sparsification is given by

$$Q_{topK} = d \cdot \phi_{global} \cdot (q + \log_2 d). \tag{8}$$

One can easily observe that for $\phi_{local} << \phi_{global}$, Q_{TCS} is smaller than Q_{topK}, especially when q is small.

B. Error accumulation

Due to sparsification, there is an error in the local model difference sent to the PS by client n, which can be expressed as:

$$\mathbf{e}_{n,t} = \Delta\boldsymbol{\theta}_{n,t} \otimes (1 - \mathbf{m}_{global} - \mathbf{m}^n_{local}). \tag{9}$$

It has been shown that the convergence speed can be improved by propagating the current compression error to next iterations [19], [25], [26]. That is, at iteration t, client n intends to send

$$\bar{\Delta}\boldsymbol{\theta}_{n,t} = \Delta\boldsymbol{\theta}_{n,t} + \mathbf{e}_{n,t-1} \tag{10}$$

Algorithm 2 TCS with error accumulation

```
1:  for $t = 1, \dots, T$ do
2:      Client side:
3:      for $n = 1, \dots, N$ do in parallel
4:          Receive $\Delta\boldsymbol{\theta}_{t-1}$ from PS
5:          $\mathbf{m}_{global} = S_{top}(\Delta\boldsymbol{\theta}_{t-1}, K_{global})$
6:          Update model: $\boldsymbol{\theta}_{n,t} = \boldsymbol{\theta}_{n,t-1} + \Delta\boldsymbol{\theta}_{t-1}$
7:          Perform $H$ local updates and compute $\Delta\boldsymbol{\theta}_{n,t}$
8:          Error Feedback: $\bar{\Delta}\boldsymbol{\theta}_{n,t} = \Delta\boldsymbol{\theta}_{n,t} + \mathbf{e}_{n,t-1}$
9:          $\mathbf{m}^{n}_{local} = S_{top}(\bar{\Delta}\boldsymbol{\theta}_{n,t} \otimes (1 - \mathbf{m}_{global}), K_{local})$
10:         $\widetilde{\Delta}\boldsymbol{\theta}_{n,t} = (\mathbf{m}^{n}_{local} + \mathbf{m}_{global}) \otimes \bar{\Delta}\boldsymbol{\theta}_{n,t}$
11:         Send $\widetilde{\Delta}\boldsymbol{\theta}_{n,t}$ to PS
12:         $\mathbf{e}_{n,t} = \bar{\Delta}\boldsymbol{\theta}_{n,t} - \widetilde{\Delta}\boldsymbol{\theta}_{n,t}$
13:     PS side:
14:     $\Delta\boldsymbol{\theta}_{t} = \frac{1}{N}\sum_{n\in[N]} \widetilde{\Delta}\boldsymbol{\theta}_{n,t}$
15:     Send $\Delta\boldsymbol{\theta}_{t}$ to clients
```

to the PS. Accordingly, each client performs sparsification on $\bar{\Delta}\boldsymbol{\theta}_{n,t}$ instead of $\Delta\boldsymbol{\theta}_{n,t}$. The overall TCS algorithm with error accumulation is summarized in Algorithm 2.

We note that $S_{top}(\mathbf{v}, K)$ in Algorithm 2 maps vector $\mathbf{v} \in \mathbb{R}^d$ to a mask vector $\mathbf{m} \in \{0,1\}^d$, such that if $\overline{v}_K$ is the Kth greatest value in $|\mathbf{v}|$, then $\mathbf{m}_i = 1$ if $|\mathbf{v}_i| \geq \overline{v}_K$ and 0 otherwise.

C. *Quantization framework*

For the quantization, we consider a similar approach to the scaled sign operator [6], [19] which, for given d dimensional vector $\mathbf{u}$, maps the value of ith parameter, $\mathbf{u}_i$, to a quantized scalar value, $\mathcal{Q}(\mathbf{u}_i)$, in the following way:

$$\mathcal{Q}(\mathbf{u}_i) = \frac{||\mathbf{u}||_1}{d}\text{sign}(\mathbf{u}_i), \tag{11}$$

where sign$(\cdot)$ is the sign operator. It has been shown that the impact of the quantization error can be reduced by dividing $\mathbf{u}$ into P smaller disjoint blocks $\{\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(P)}\}$, and then applying quantization to each block separately, and often these blocks correspond to layers of DNN architecture [6]. Hence, we use a variation of the scaled sign operator that utilizes multiple mean values. Let u_{max} and u_{min} be the maximum and minimum values in vector $|\mathbf{u}|$, respectively. We divide the interval $[u_{max}, u_{min}]$ into P disjoint intervals $I_1, \dots, I_P$, such that the pth interval, I_p, is given as $I_p = [\sigma^{p-1}u_{max}, \sigma^{p}u_{max}]$ where, $\sigma = \left(\frac{u_{min}}{u_{max}}\right)^{1/P}$. Further, let μ_p be the average of the values assigned to interval I_p. Then, fractional quantization maps the value of the ith parameter, $\mathbf{u}_i$, to a quantized scalar value, $\mathcal{Q}_f(\mathbf{u}_i)$, in the following way:

$$\mathcal{Q}_f(\mathbf{u}_i) = \sum_{p=1}^{P} \mathbb{1}_{\{\mathbf{u}_i \in I_p\}} \mu_p \text{sign}(\mathbf{u}_i), \tag{12}$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function. Since there are P intervals in total, $\log_2 P$ bits are sufficient to identify the corresponding interval of $\mathbf{u}_i$, $i = 1, \dots, d$. An additional bit is sufficient to represent the sign; hence, in total, $\log_2 P + 1$ bits are required per parameter. Additional $32 \cdot P$ bits are required to convey the mean values of the intervals. Consequently, for a given d dimensional vector $\mathbf{u}$, a total of $d(\log_2 P + 1) + 32 \cdot P$ bits are required, where the second term is often negligible.

III. Numerical Results

A. *Simulation Setup*

To evaluate the performance of the proposed TCS strategy, we consider the image classification task on the CIFAR-10 dataset [27], which consists of 10 image classes, organized into 50K training and 10K test images, respectively. We employ the ResNet-18 architecture as the DNN [28], which consists of 8 basic blocks, each with two 3x3 convolutional layers and batch normalization and contains 11,173,962 trainable parameters in total. We consider a network of $N = 10$ clients and a federated setup, in which the training dataset is divided among the clients in a disjoint manner. The images, based on their classes, are distributed in an identically and independently distributed (IID) manner among the clients.

B. *Implementation*

For performance evaluation, we consider the centralized training as our main benchmark, where we assume that all the training dataset is collected at one client. We set the batch size to 128 and the learning rate to $\eta = 0.1$. The performance of this centralized setting will be referred to as the "Baseline" in our simulation results. For all the FL strategies considered in this work we set the batch size to 64, and adopt the linear learning rate scaling rule in [29], where the learning rate is scaled according to the cumulative batch size and the total number of samples trained by all the clients, taking the batch size of 128 as a reference value with the corresponding learning rate $\eta = 0.1$. Hence, for our setup with $N = 10$ clients, we use the learning rate $\eta = 0.5$. Further, in all the FL implementations we employ the warm up strategy [29], where the learning rate is initially set to $\eta = 0.1$, and is increased to its corresponding scaled value gradually in the first 5 epochs. We also note that during the warm up phase we do not employ sparsification and quantization methods for communication.

The DNN architecture is trained for 300 epochs and the learning rate is reduced by a factor of 10 after the first 150 and 225 epochs, respectively [28], [30]. Lastly, in all the simulations we employ L2 regularization with a given weight decay parameter 10^{-4}.

For performance evaluation, we consider the top-K sparsification scheme as a second benchmark. For top-K sparsification we set the sparsification ratio to $\phi = 10^{-2}$. Accordingly, for the proposed TCS strategy we set $\phi_{global} = 10^{-2}$ and $\phi_{local} = 10^{-3}$. We want to emphasize that for the TCS strategy with 10 clients, these parameters imply a maximum of 0.02 sparsification ratio; in other words $\times 50$ compression in the PS-to-client direction as well, which is not the case for top-K sparsification.

We recall that one of the key design parameters of FL is the number of local steps H. Hence, we use TCS-LH to denote the TCS scheme with H local iterations. We use only TCS to

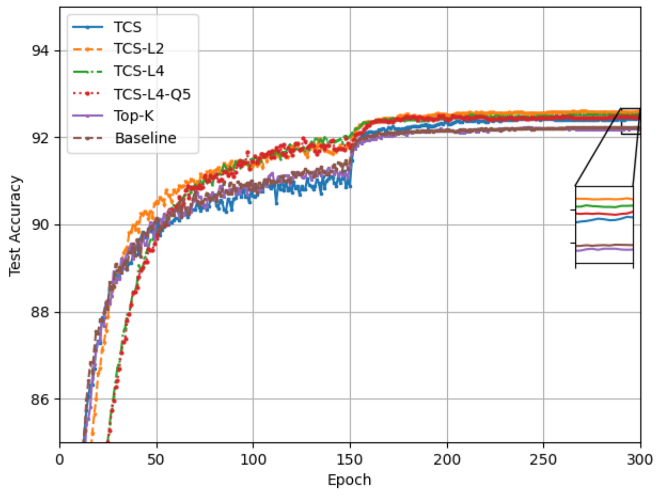

Fig. 1: Comparison of the test accuracy results for variants of the proposed TCS strategy with top-K sparsification and the centralized baseline over 300 epochs (for $\eta = 0.5$).

refer to the FedSGD scheme with $H = 1$. For FedAvg, we consider $H = 2$ in our simulations. Finally, we also employ quantization strategy to represent each non-zero value with $q << 32$ bits and use the notation 'Qq' to denote the number of bits used to represent each element. For example, TCS-L4-Q5 denotes the TCS strategy with $H = 4$ local iterations along with 5-bit quantization.

For performance evaluation, we employ two performance metrics: *test accuracy* and the *bit budget*, corresponding to the performance of the final trained model and the communication load, respectively. More specifically, the bit budget refers to the average number of bits conveyed from a client to the PS per parameter per iteration.

C. Simulation Results

In our simulation, we consider 6 schemes, namely the Baseline, top-K sparsification, TCS, TCS-L2, TCS-L4, and TCS-L4-Q5, where the first two are used as benchmark schemes. For each scheme we take the average over 5 trials. The final test accuracy results, with mean and standard deviation, and the bit budget for each scheme is presented in Table I. In Figure 1, we present the test accuracy results with respect to the epoch index.

We observe that the proposed TCS scheme requires 12% lower bit budget than top-K sparsification while achieving a higher average test accuracy. Similarly, it achieves approximately $\times 100$ reduction in the communication load without losing the accuracy. We also observe that TCS with multiple local iterations, in particular, TCS-L2 and TCS-L4, achieve higher test accuracy compared to TCS with single local iteration. While the best accuracy is achieved by TCS-L2, almost the same accuracy is achieved by TCS-L4, but with half the average bit budget. Although this may seem counter-intuitive at a first glance, we remark that due to random batch sampling

Method	Test Accuracy (mean $\pm$ std)	Bit budget
TCS	92.44 $\pm$ 0.143	0.363
TCS-L2	**92.578 $\pm$ 0.189**	0.1815
TCS-L4	92.53 $\pm$ 0.22	0.0907
TCS-L4-Q5	92.485 $\pm$ 0.22	0.01675
Top-K	92.194 $\pm$ 0.247	0.41
Baseline	92.228 $\pm$ 0.232	-

TABLE I: Test accuracy (for $\eta = 0.5$) and bit budget comparison between the studied schemes. While best test accuracy is achieved with TCS with 2 local iterations, TCS with 4 local iterations can achieve a similar test accuracy with only half the communication load. The communication load of TCS can be further reduced with quantization at the expense of a very small reduction in the test accuracy.

in SGD, gradient values behave as random variables; and hence, using model difference over H iterations may provide a more accurate observation to be able to identify new important weights. To reduce the bit budget further we consider TCS-L4-Q5, where all the non-zero values are represented with 5 bits in total while one bit is used for the sign. We observe that TCS with quantization can achieve $\times 2000$ reduction in the communication load, with an even better test accuracy compared to the centralized baseline.

We emphasize that the impact of quantization on the bit budget is more visible with TCS compared to top-K sparsification. When quantization is used with top-K sparsification, the number of bits used for the location becomes the bottleneck as quantization cannot reduce that. When TCS (with $\phi_{global} = 10^{-2}$ and $\phi_{local} = 10^{-3}$) is employed together with 5-bit quantization, the corresponding bit budget is 0.067 (bits per element). On the other hand, top-K sparsification ($\phi = 10^{-2}$) with 5-bit quantization requires a bit budget of 0.14, which is more than twice the bit budget of TCS. Furthermore, when the number of bits used for quantization decreases, TCS becomes more and more communication efficient.

IV. Conclusion

In this paper, we introduced a novel sparse communication strategy for communication-efficient FL, called *time correlated sparsification (TCS)*, by establishing an analogy between network pruning and gradient sparsification frameworks, and benefiting from the side information available at the PS to reduce the communication load from the clients to the PS. The proposed strategy is built upon the assumption that at "important locations" the model difference (or the gradient) changes slowly over time, and utilizes this correlation over iterations to reduce the communication load. Through extensive simulations on CIFAR-10 dataset, we show that TCS can meet or even surpass the centralized baseline accuracy with $\times 100$ sparsification, and can reach up to $\times 2000$ reduction in the communication load when it is employed together with quantization. The proposed TCS strategy results in the sparsification of the model updates transmitted from the PS to the clients as well, which can further be exploited to reduces the communication load in the downlink direction.

REFERENCES

[1] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, and A. Feng, “Privacy-preserving federated brain tumour segmentation,” in *Machine Learning in Medical Imaging*. Springer International Publishing, 2019, pp. 133–141.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 54, Fort Lauderdale, FL, USA, 20–22 Apr 2017, pp. 1273–1282.

[3] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, “Hierarchical federated learning across heterogeneous cellular networks,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8866–8870.

[4] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, “Adaptive federated learning in resource constrained edge computing systems,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[5] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. [Online]. Available: https://arxiv.org/abs/1610.05492

[6] S. Zheng, Z. Huang, and J. Kwok, “Communication-efficient distributed blockwise momentum SGD with error-feedback,” in *Advances in Neural Information Processing Systems 32*, 2019, pp. 11 450–11 460.

[7] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, “QSGD: Communication-efficient SGD via gradient quantization and encoding,” in *Advances in Neural Information Processing Systems 30*, 2017, pp. 1709–1720.

[8] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, “signSGD: Compressed optimisation for non-convex problems,” in *Proceedings of the 35th International Conference on Machine Learning*, Stockholmsmässan, Stockholm Sweden, Jul 2018, pp. 560–569.

[9] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, “1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns,” in *Interspeech 2014*, September 2014.

[10] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, “Sparsified SGD with memory,” in *Advances in Neural Information Processing Systems 31*, 2018, pp. 4448–4459.

[11] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, “The convergence of sparsified gradient methods,” in *Advances in Neural Information Processing Systems 31*, 2018, pp. 5976–5986.

[12] J. Wangni, J. Wang, J. Liu, and T. Zhang, “Gradient sparsification for communication-efficient distributed optimization,” in *Advances in Neural Information Processing Systems 31*, 2018, pp. 1305–1315.

[13] A. F. Aji and K. Heafield, “Sparse communication for distributed gradient descent,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 440–445.

[14] N. F. Eghlidi and M. Jaggi, “Sparse communication for training deep networks,” *CoRR*, vol. abs/2009.09271, 2020.

[15] E. Ozfatura, K. Ozfatura, and D. Gunduz, “Time-correlated sparsification for communication-efficient federated learning,” *CoRR*, vol. abs/2101.08837, 2021.

[16] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi, “Dynamic model pruning with feedback,” in *International Conference on Learning Representations*, 2020.

[17] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” in *International Conference on Learning Representations*, 2019.

[18] T. M. Cover and J. A. Thomas, *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, July 2006.

[19] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, “Error feedback fixes SignSGD and other gradient compression schemes,” in *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, California, USA, Jun 2019, pp. 3252–3261.

[20] M. M. Amiri and D. Gunduz, “Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 2155–2169, 2020.

[21] G. Zhu, Y. Wang, and K. Huang, “Broadband analog aggregation for low-latency federated edge learning,” *IEEE Trans. Wireless Comms.*, 2019.

[22] T. Sery and K. Cohen, “On analog gradient descent learning over multiple access fading channels,” *IEEE Trans. Signal Proc.*, vol. 68, pp. 2897–2911, 2020.

[23] M. M. Amiri and D. Gündüz, “Federated learning over wireless fading channels,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3546–3557, 2020.

[24] G. Zhu, Y. Du, D. Gündüz, and K. Huang, “One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis,” *IEEE Transactions on Wireless Communications*, pp. 1–1, 2020.

[25] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, “1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns,” in *Interspeech 2014*, September 2014.

[26] J. Wu, W. Huang, J. Huang, and T. Zhang, “Error compensated quantized SGD and its applications to large-scale distributed optimization,” in *Proceedings of the 35th International Conference on Machine Learning*, Stockholmsmässan, Stockholm Sweden, Jul 2018, pp. 5325–5333.

[27] A. Krizhevsky, V. Nair, and G. Hinton, “Cifar-10 (canadian institute for advanced research).” [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[28] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 630–645.

[29] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, “Accurate, large minibatch SGD: training imagenet in 1 hour,” *CoRR*, vol. abs/1706.02677, 2017.

[30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.