Communication-efficient Subspace Methods for High-dimensional Federated Learning

Zai Shi
Department of Electrical and Computer Engineering
The Ohio State University
Columbus, USA
shi.960@osu.edu

Atilla Eryilmaz
Department of Electrical and Computer Engineering
The Ohio State University
Columbus, USA
eryilmaz.2@osu.edu

***Abstract*—As an emerging technique to employ machine learning processes within an edge computing infrastructure, federated learning (FL) has aroused great interests in both industry and academia. In this paper, we consider a potential challenge of FL in a wireless setup, whereby uplink communication from edge devices to the central server has limited capacity. This is particularly important for machine learning tasks (such as training deep neural networks) in FL with extremely high-dimensional domains that can substantially increase the communication burden. To tackle this challenge, we first propose a basic method called Subspace Stochastic Gradient Descent for Federated Learning (FL-SSGD) to introduce the idea of subspace methods. Through theoretical analysis, we show that by choosing appropriate subspace matrices in FL-SSGD, we can reduce uplink communication costs compared to classical FedAvg method. To improve FL-SSGD, we then propose another method called Subspace Stochastic Variance Reduced Gradient for Federated Learning (FL-SSVRG) that has a faster convergence rate with less assumptions on objective functions. By conducting experiments of a nonconvex machine learning problem in two FL setups, we demonstrate the advantages of our methods compared to other communication-efficient methods.**

***Index Terms*—Federated Learning, Communication Efficiency, Subspace Methods**

I. Introduction

With the rapid development of deep learning and mobile computing, federated learning (FL) has emerged as a new technique to utilize edge devices for an acceleration of machine learning tasks. There are many scenarios and applications where FL can be useful. For example, to support the typing prediction feature provided by its keyboard app, a company can set up a server deployed with FL algorithms, which cooperates with its users to learn a model of their typing behavior without exchanging private data.

Figure 1 illustrates a common process of FL. In this process, the central server sends a common update ΔW to all the devices. Based on this update, each device produces its own update $\Delta W_i(D_i)$ based on its private dataset D_i and then sends it back to the central server. The central server hopes to obtain an accurate machine learning model after several rounds of this process.

This research is funded in part by NSF grants: CNS-NeTS-1717045, CNS-SpecEES-1824337, CNS-NeTS-2007231,CNS-NeTS-2106679, IIS-2112471; and the ONR Grant N00014-19-1-2621.

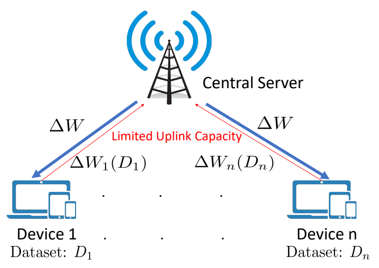


Fig. 1: A typical model of FL with limited uplink capacity

Among previous works, the most classical FL algorithm is called Federated Averaging (FedAvg) method [1], which we can use to exemplify this process. Suppose that the central server wants to learn a model parameter x by solving $\min_{x\in\mathbb{R}^d}\sum_{i=1}^n w_i f_i(x)$, where f_i is the loss function related to Device i's private dataset and w_i is the weight assigned to f_i. In one iteration of FedAvg, the server sends the current learned parameter x_t to each device, and then each device sends back a stochastic gradient $g_i(x_t)$ of f_i at x_t. The central server receives all $g_i(x_t)$'s and updates the model parameter as $x_{t+1} = x_t - \alpha\sum_{i=1}^n w_i g_i(x_t)$, where α is the step size.

In many machine learning problems, the dimension of the model parameter x can be very high, such as in deep neural networks [2]. Then communicated vectors, like stochastic gradients in FedAvg, may have high dimensions as well, which leads to large communication costs. This situation is more crucial in a wireless setup of FL as shown in Figure 1, where each uplink channel has limited capacity due to the power limitation of mobile devices and interference from other devices.

This motivates us in this paper to focus on the challenge of limited uplink capacity in FL solving high-dimensional nonconvex problems. The contributions of our paper are as follows.

- We first propose a basic method called *Subspace Stochastic Gradient Descent for Federated Learning (FL-SSGD)* to introduce the idea of subspace methods for uplink communication time-reduction in Section III. We give a theoretical analysis of its performance and show that FL-SSGD can reduce the uplink communication time of FedAvg while achieving the same order of convergence rate.

- We then propose an improved method called *Subspace Stochastic Variance Reduced Gradient for Federated Learning (FL-SSVRG)* in Section IV, which achieves a faster convergence rate while sharing the benefits of FL-SSGD for uplink communication time-reduction. Besides, it relaxes the bounded gradient assumption needed by FL-SSGD.
- Through experiments of a machine learning problem in two FL setups, we demonstrate the advantages of our methods compared to other benchmark communication-efficient methods in Section V.

Next, we will discuss some related works to our paper, including literature on subspace methods and gradient sparsification methods.

A. Related Works

1) Subspace Methods: Our methods are based on stochastic subspace descent (SSD) proposed in [3]. SSD is a generalization of coordinate descent schemes (see Section 1.1 of [3]) by projecting the gradient onto a random subspace (not necessarily a directional gradient) at each iteration. The authors proved convergence rates of the basic method for convex problems and then proposed an improved method for strong-convex functions. To get even faster convergence rates, the authors in [4] proposed a new randomized second-order optimization algorithm, called Stochastic Subspace Cubic Newton (SSCN), by using subspace of Hessian matrices and gradient vectors. But their convergence results are still for convex problems. In contrast, our paper extends subspace methods to FL setup and proves convergence rates for nonconvex problems.

2) Gradient Sparsification Methods: Gradient sparsification is an alternative method that can be used for communication time-reduction in FL. In each iteration of this method, each device compresses its transmitted value before communication, and stores errors brought by compression, which will be used for compensation in the next iteration. Because of space limitation, we only mention related works for nonconvex optimization problems with theoretical convergence results in the following.

There are two kinds of popular gradient compressors in this scheme. The first is called top-k compressor [5], [6], which selects top k elements of gradients in terms of magnitude. In [5], the authors proposed a basic method using this compressor and proved convergence results for convex and nonconvex problems. In [6], the authors provided the convergence results of a more efficient method with this compressor. However, the results of these two works need an artificial assumption (Assumption 1 of [5] and [6]), respectively, which is not common in optimization literature, and can be only verified by experiments.

The second compressor, called random compressor [7], [8], is more similar to ours. It randomly selects k elements from gradients. In [7], the authors proposed a method called DoubleSqueeze for nonconvex problems, which applies error compensation both to the devices and the central server. In [8], Nesterov's acceleration scheme was used in their works to accelerate convergence rates of previous methods for convex and nonconvex problems. Both of these works assumed bounded variance of stochastic gradients for their results, which is not needed in our second method.

The major difference between our methods and gradient sparsification is that we do not need error compensation for our convergence guarantees, which can spare much space for edge devices with limited storage. Meanwhile for time horizon T, our second method has a faster convergence rate $O(1/T)$ for nonconvex problems with fewer assumptions than the above works of gradient sparsification which all achieve $O(1/\sqrt{T})$ convergence rates.

B. Notational Convention

In this paper, we use $||\cdot||$ to represent l_2-norm. e_l is a basis whose l^{th} element is 1 and the others are 0. $\mathbf{I}_d$ is a $d \times d$ identity matrix. $\nabla F(x;\xi)$ represents the gradient of F with regard to x with parameters ξ. Finally, we denote the transpose of a matrix A as A^T.

II. Problem Formulation

We consider a federated learning setup with n wireless devices and one central server. The objective is to

$$\min_{x\in\mathbb{R}^d} f(x) := \sum_{i=1}^{n} w_i f_i(x), \tag{1}$$

where f_i is the function only known by Device i and possibly nonconvex, w_i is the weight assigned to f_i and d is the dimension of x. The transmitted data is assumed to be perfect without errors due to quantization, coding, channels and so on. As a common target of nonconvex optimization methods [9], we aim to obtain a first-order stationary point x^* of (1), where $||\nabla f(x^*)|| = 0$.

In this paper, we consider two situations that often arise in machine learning problems. First, f_i is assumed to take the form of $\frac{1}{m_i}\sum_{j=1}^{m_i} F(x;\xi_{i,j})$, where $\{\xi_{i,j}\}_{j=1}^{m_i}$ are m_i data points stored in Device i, as D_i shown in Figure 1. This form captures the characteristics of empirical loss problems [10] in machine learning, where F is the loss function between the real dataset $\{\xi_{i,j}\}_{j=1}^{m_i}$ and the predicted model parameterized by x.

Second, we assume d to be very large, which can be seen in machine learning problems like regression with a large feature space [10] or deep neural networks [2]. Meanwhile, we assume that uplink capacity from each device to the central server is limited due to the power limitation of edge devices and interference from other devices. It means that it may take a very long time for a device to send a d-dimensional vector like x or $\nabla f_i(x)$ to the central server, which results in large communication costs [1]. FedAvg [1] introduced in Section I is one example suffering from this problem.

[1]For downlink communication, we assume that it has sufficient capacity due to the superior performance of the central server, thus is not the focus of this paper.

Algorithm 1 FL-SSGD

```
1: Input of Server: Initial value $x_0$, step size $\alpha$, time horizon $T$
2: for $t = 0$ to $T - 1$ do
3:     Server: Sample a matrix $P_{t,i} \in \mathbb{R}^{d\times\tau_{t,i}}$ independent and identically distributed (i.i.d.) with regard to previous iterations and satisfying Assumption 4, where $\tau_{t,i} \leq d$. Send $x_t, P_{t,i}$ to Device $i, \forall i$.
4:     Device $i$, $\forall i$: Randomly draw one sample $\Xi_{t,i}$ from its dataset and compute the stochastic gradient $\nabla F(x_t; \Xi_{t,i})$. Send $h_{t,i} = P_{t,i}^T \nabla F(x_t; \Xi_{t,i})$ to the server.
5:     Server: Receive information from all the devices and compute $g_t = \sum_{i=1}^n \frac{w_i d}{l_i} P_{t,i}(P_{t,i}^T P_{t,i})^{-1} h_{t,i}$, where $l_i$ is defined in Assumption 4. Update the model parameters as $x_{t+1} = x_t - \alpha g_t$
6: end for
```

For this high-dimensional setup, we hope to propose an efficient algorithm to minimize uplink communication cost with a convergence guarantee for solving (1). To that end, we will first discuss an intuitive method without extra space requirements needed in gradient sparsification methods. Then we will improve its performance using additional techniques.

III. Basic Subspace Method: FL-SSGD

An intuitive strategy to deal with the high-dimensional setup is that each device only sends a subset of dimensions determined by its uplink capacity. Based on this intuition and FedAvg discussed in Section I, we propose the following algorithm called Subspace Stochastic Gradient for Federated Learning (FL-SSGD) shown in Algorithm 1.

In this algorithm, instead of stochastic gradients $\nabla F(x_t; \Xi_{t,i})$ in d dimension, each device sends $P_{t,i}^T \nabla F(x_t; \Xi_{t,i})$ in $\tau_{t,i}$ dimensions, and $\tau_{t,i}$ can be chosen flexibly according to its uplink rate, data size and other factors. Here $P_{t,i} \in \mathbb{R}^{d\times\tau_{t,i}}$ is a subspace matrix to reduce dimensions. If $P_{t,i} = \mathbf{I}_d$, then FL-SSGD becomes FedAvg. Now the question becomes how to generate $P_{t,i}$ that guarantees the convergence of FL-SSGD with uplink communication time-reduction. To answer this question, we first need the following assumptions for FL-SSGD.

Assumption 1. *There is a constant $G > 0$ s.t. $\mathbb{E}||\nabla F(x; \Xi_{t,i})|| \leq G$ for any x. Here the expectation is taken with regard to $\Xi_{t,i}$ in FL-SSGD.*

Assumption 2. *$F(\cdot; \xi)$ is L-smooth for any ξ, i.e., $||\nabla F(x; \xi) - \nabla F(y; \xi)|| \leq L||x - y||$ for any x and y.*

Assumption 3. *$f^* := \inf_x f(x) < \infty$*

Assumption 4. *$\mathbb{E}[P_{t,i}(P_{t,i}^T P_{t,i})^{-1} P_{t,i}^T] = \frac{l_i}{d}\mathbf{I}_d$ for some $0 < l_i \leq d$. Here $P_{t,i}$ and $P_{t,i}(P_{t,i}^T P_{t,i})^{-1} P_{t,i}^T$ are regarded as $d \times d$ zero matrices if $\tau_{t,i} = 0$.*

It is noted that Assumptions 1-3 are common in optimization literature. Compared with the requirements of subspace matrices in [3] (Assumption 2.1 of [3]), Assumption 4 is more general by allowing $P_{t,i}$ to be a zero matrix for some t, which makes l_i to be any positive real number less than d instead of a positive integer in [3]. Obviously, examples of subspace matrices mentioned in [3] can be used in our method. Meanwhile, we have a simple way to construct a $P_{t,i}$ with a certain l_i as follows.

Remark 1.
- *One example of $P_{t,i}$ satisfying Assumption 4 with $l_i \in \mathbb{Z}^+$: First generate a set $\mathcal{A}$ by randomly selecting $\tau_{t,i} \geq 1$ numbers out of $\{1, ..., d\}$. Then let the jth column of $P_{t,i} \in \mathbb{R}^{d\times\tau_{t,i}}$ be d-dimensional $e_{\mathcal{A}(j)}$, $\forall j \in \{1, ..., \tau_{t,i}\}$, where $\mathcal{A}(j)$ is jth element of $\mathcal{A}$. Now $l_i = \tau_{t,i}$ in this example.*
- *One example of $P_{t,i}$ satisfying Assumption 4 with $l_i \in \mathbb{R}^+$: If we use the above procedure to generate $P_{t,i}$ with $\tau_{t,i} = d_1$ in probability q and $\tau_{t,i} = d_1 - 1$ ($P_{t,i}$ is a zero matrix if $\tau_{t,i} = 0$) in probability $1 - q$, then $l_i = qd_1 + (1 - q)(d_1 - 1)$, which can be any positive real number by choosing q and d_1.*

In this paper, we assume that l_i is time-invariant for simplicity. It is easy to extend our results to a time-variant case, which can be used for time-variant uplink capacity. From Remark 1, we can see that $P_{t,i}$ can be determined and represented by $\mathcal{A}$, which only involves $\tau_{t,i}$ integers with small space complexity, thus has little impact on downlink communication cost in FL-SSGD. Now we can turn to the convergence result of FL-SSGD.

Theorem 1. *If Assumptions 1-4 are satisfied, then Algorithm 1 with $\alpha = 1/\sqrt{T}$ satisfies*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}||\nabla f(x_t)||^2 \leq \frac{f(x_0) - f^*}{\sqrt{T}} + \frac{dLn}{2\sqrt{T}}\sum_{i=1}^n \frac{w_i^2}{l_i} G^2 \quad (2)$$

Proof. By Assumption 2, we have f is L-smooth and

$$f(x_{t+1}) \leq f(x_t) + \nabla f(x_t)^T(x_{t+1} - x_t) + \frac{L}{2}||x_{t+1} - x_t||^2$$
$$\leq f(x_t) - \alpha\nabla f(x_t)^T g_t + \frac{\alpha^2 L}{2}||g_t||^2. \quad (3)$$

where (3) is from procedure of FL-SSGD. From Assumption 4, we have

$$\mathbb{E}[g_t|x_t] = \nabla f(x_t)$$
$$\mathbb{E}[||P_{t,i}(P_{t,i}^T P_{t,i})^{-1} P_{t,i}^T \nabla F(x_t; \Xi_{t,i})||^2|x_t] = \frac{l_i}{d}\mathbb{E}[||\nabla F(x_t; \Xi_{t,i})||^2|x_t]$$

Now taking expectation to both sides of (3) conditioned on x_t, we have

$$\mathbb{E}[f(x_{t+1})|x_t] \leq f(x_t) - \alpha||\nabla f(x_t)||^2$$
$$+ \frac{\alpha^2 d^2 Ln}{2}\sum_{i=1}^n \frac{w_i^2}{l_i^2}\mathbb{E}[||P_{t,i}(P_{t,i}^T P_{t,i})^{-1} P_{t,i}^T \nabla F(x_t; \Xi_{t,i})||^2|x_t]$$
$$= f(x_t) - \alpha||\nabla f(x_t)||^2 + \frac{\alpha^2 dLn}{2}\sum_{i=1}^n \frac{w_i^2}{l_i} G^2$$

where the first inequality is from $||\sum_{i=1}^n a_i||^2 \leq n\sum_{i=1}^n ||a_i||^2$ and the second is from Assumption 1. Telescoping from $t=0$ to $T-1$, taking expectation with regard to all previous iterations and using Assumption 3, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}||\nabla f(x_t)||^2 \leq \frac{f(x_0)-f^*}{T\alpha} + \frac{\alpha dLn}{2}\sum_{i=1}^n \frac{w_i^2}{l_i}G^2$$

If $\alpha = 1/\sqrt{T}$, we can have the final result. □

The above metric is also seen in [6]. From the theorem, we can see that FL-SSGD with time horizon T can achieve an $O(1/\sqrt{T})$ convergence rate, which is the same with FedAvg (easy to see by letting $l_i = d$), but has a larger constant term due to compression. Meanwhile, unlike gradient sparsification methods introduced in Section I-A2, FL-SSGD does not need edge devices to store their own error accumulation in d-dimension due to sparsification. Therefore, the storage complexity of FL-SSGD is constant while it is $O(nd)$ for gradient sparsification methods. The latter is not preferred in our high-dimensional setup when edge devices like mobile phones have limited storage.

Meanwhile, for Assumption 4, we have the following lemma (Lemma 5.2 of [4]).

Lemma 1. *If Assumption 4 is satisfied, then* $\mathbb{E}[\tau_{t,i}] = l_i$.

The above lemma tells us that in the long run, each device "seems" to transmit l_i dimensions in each iteration of FL-SSGD. From Theorem 1 and Lemma 1, we can see that a smaller l_i leads to a slower convergence rate, but also less transmission data in the uplink. Therefore, we need to discuss how to choose l_i to minimize the uplink communication time in FL-SSGD. We will take advantage of the bound in Theorem 1, which can represent the worst-case performance of this algorithm. In the following, we will consider two transmission schemes.

A. Time Sharing

First suppose that only one device is allowed to transmit at any time of uplink communication and the rate of Device i is r_i. This case can serve as an upper bound of other cases where at most $b > 1$ devices are allowed to transmit concurrently. Also suppose that we need B bits to represent each dimension of a vector accurately enough. Then, the expected uplink communication time for one iteration of FL-SSGD is Bl_i/r_i for Device i. To achieve $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}||\nabla f(x_t)||^2 \leq \varepsilon$, the expected uplink communication time in total is bounded by

$$\left(\sum_{i=1}^n \frac{Bl_i}{r_i}\right)\left(\frac{f(x_0)-f^*}{\varepsilon} + \frac{dLn}{2\varepsilon}\sum_{i=1}^n \frac{w_i^2}{l_i}G^2\right)^2. \quad (4)$$

Now we can find the best l_i by minimizing the above bound. Since FL-SSGD with $l_i = d$ is equivalent to FedAvg, the above bound can also be applied to FedAvg. Due to the first term of (4), we can see that l_i cannot be too large, which means that FedAvg is not always the best algorithm for uplink communication time minimization. Due to the second term, l_i cannot be too small as well.

B. Channel Sharing

Now, we turn to the discussion of the case where all the devices share the uplink channel with capacity C. To achieve $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}||\nabla f(x_t)||^2 \leq \varepsilon$, we should choose l_i and r_i by solving

$$\min_{0<l_i\leq d, r_i>0}\left(\frac{f(x_0)-f^*}{\varepsilon} + \frac{dLn}{2\varepsilon}\sum_{i=1}^n \frac{w_i^2}{l_i}G^2\right)^2 \mathbb{E}\max_{i\in 1,\dots,n}\frac{B\tau_{0,i}}{r_i}$$
$$s.t. \sum_{i=1}^n r_i \leq C$$

where $\mathbb{E}\max_{i\in 1,\dots,n}\frac{B\tau_{0,i}}{r_i}$ is expected uplink communication time per iteration due to synchronization, and the expectation is taken with regard to $\{\tau_{0,1},\dots,\tau_{0,n}\}$. To facilitate the solution to this problem, we choose l_i to be integer and $\tau_{t,i}$ to be deterministic, which is equal to l_i (see Remark 1). Then, we can rewrite the optimization as

$$\min_{l_i\leq d, r_i>0}\left(\frac{f(x_0)-f^*}{\varepsilon} + \frac{dLn}{2\varepsilon}\sum_{i=1}^n \frac{w_i^2}{l_i}G^2\right)^2 \max_{i\in 1,\dots,n}\frac{Bl_i}{r_i}$$
$$s.t. \sum_{i=1}^n r_i \leq C, l_i \in \mathbb{Z}^+, \forall i.$$

Meanwhile, for any l_i, we can get $r^* = Cl_i/\sum_{i=1}^n l_i$ by minimizing $\max_{i\in 1,\dots,n}\frac{Bl_i}{r_i}$ because r_i only appears in this term. Accordingly, the problem becomes

$$\min_{l_i\in\mathbb{Z}^+, l_i\leq d}\left(\frac{f(x_0)-f^*}{\varepsilon} + \frac{dLn}{2\varepsilon}\sum_{i=1}^n \frac{w_i^2}{l_i}G^2\right)^2 \frac{B\sum_{i=1}^n l_i}{C}, \quad (5)$$

which is much easier to solve. Again, $l_i = d$ is not always the best solution due to the first term of (5), thus FL-SSGD can outperform FedAvg for uplink communication time.

IV. Improved Method: FL-SSVRG

In this section, we will propose a method that has a faster convergence rate than FL-SSGD with fewer assumptions. The method, called Subspace Stochastic Variance Reduction Gradient for Federated Learning (FL-SSVRG), is based on a variance reduction strategy introduced in SVRG [11]. In fact, there is already a work combining SVRG with a subspace method for a faster convergence rate (See Algorithm 2.3 of [3]). The main shortcoming of this work for our setup is that in the beginning of each epoch, their method still needs full-dimensional gradients, which can have large communication costs. Meanwhile, their work only gave convergence results for strongly-convex objective functions. In contrast, our method can reduce uplink communication time in each iteration with theoretical convergence results for nonconvex problems.

FL-SSVRG is an epoch-based method shown in Algorithm 2. Here x_k^s represents the value at iteration k of epoch s. At the start of each epoch ($k = 0$), each device computes its full gradient $\nabla f_i(x_k^s)$ and uses the same subspace matrix P_k^s with other devices for communication cost reduction. In

Algorithm 2 FL-SSVRG

```
1: Input of Server: Initial value $x_0^0$, step size $\alpha$, inner loop iterations $K$, outer loop iterations $S$.
2: for $s = 0$ to $S-1$ do
3:     for $k = 0$ to $K-1$ do
4:         if $k = 0$ then
5:             Server: Sample $K$ independent matrices $\{P_0^s, ..., P_{K-1}^s\}$ where $P_k^s \in \mathbb{R}^{d\times\tau_k^s}$ satisfies Assumption 5 with $\tau_k^s \le d$. Send $x_k^s$, $\{P_0^s, ..., P_{K-1}^s\}$ to all the devices.
6:             Device $i$, $\forall i$: Draw all the samples and compute the gradient $\nabla f_i(x_0^s)$. Send $h_{0,i}^s = (P_0^s)^T \nabla f_i(x_0^s)$ to the server and store $\{h_{1,i}^s, ..., h_{K-1,i}^s\}$ where $h_{k,i}^s = (P_k^s)^T \nabla f_i(x_0^s)$.
7:             Server: Update the model parameters as $x_1^s = x_0^s - \alpha g_0^s$ with $g_0^s = \sum_{i=1}^n \frac{w_i d}{l} P_0^s((P_0^s)^T P_0^s)^{-1} h_{0,i}^s$.
8:         else
9:             Server: Sample a matrix $P_{k,i}^s \in \mathbb{R}^{d\times\tau_{k,i}^s}$ satisfying Assumption 5 and independent of previous iterations, where $\tau_{k,i}^s \le d$. Send $x_k^s, P_{k,i}^s$ to Device $i$, $\forall i$.
10:            Device $i$, $\forall i$: Randomly draw a sample $\Xi_{k,i}^s$ from its dataset, and compute two stochastic gradients $\nabla F(x_k^s; \Xi_{k,i}^s)$ and $\nabla F(x_0^s; \Xi_{k,i}^s)$. Send $v_{k,i}^s = (P_{k,i}^s)^T \left(\nabla F(x_k^s; \Xi_{k,i}^s) - \nabla F(x_0^s; \Xi_{k,i}^s)\right)$ and $h_{k,i}^s$ to the server. Delete $h_{k,i}^s$ from storage.
11:            Server: Update the model parameters as $x_{k+1}^s = x_k^s - \alpha g_k^s$ with $g_k^s = \sum_{i=1}^n w_i[\frac{d}{l_i} P_{k,i}^s((P_{k,i}^s)^T P_{k,i}^s)^{-1} v_{k,i}^s + \frac{d}{l} P_k^s((P_k^s)^T P_k^s)^{-1} h_{k,i}^s]$
12:        end if
13:    end for
14:    Server: Let $x_0^{s+1} = x_{k+1}^s$.
15: end for
```

the subsequent iterations of each epoch ($k \neq 0$), each device computes two stochastic gradients at x_k^s and x_0^s using the same sample and adopts two subspace matrices, P_k^s and $P_{k,i}^s$, for communication cost reduction. The former matrix must be the same for all the devices and the latter can be different. When $k = 0$, the number of dimensions in uplink communication is τ_0^s for each device; when $k \neq 0$ it becomes $\tau_k^s + \tau_{k,i}^s$ for Device i. These parameters can be set according to uplink rates of edge devices. The intuition behind FL-SSVRG is that the variance of g_k^s we constructed is smaller than g_t used in FL-SSGD so that a faster convergence rate can be achieved.

Before discussing its performance, we first introduce Assumption 5 that is needed for FL-SSVRG.

Assumption 5. *P_k^s and $P_{k,i}^s$ satisfy Assumption 4 for some $0 < l \le d$ and $0 < l_i \le d$, respectively.*

We continue to assume that l and l_i is time-invariant for simplicity, and it is easy to extend our results to a time-variant case. Similar to the discussion below Remark 1, if P_k^s and $P_{k,i}^s$ are generated as shown in Remark 1, they have little impact on downlink communication cost in FL-SSVRG. Next we show the convergence rate of FL-SSVRG.

Theorem 2. *Define a positive sequence $\{c_k\}_{k=0}^K$:*

$$c_k = c_{k+1} + \alpha c_{k+1} + (\frac{\alpha^2 L}{2} + \alpha^2 c_{k+1})(\frac{4dL^2}{l} + \sum_{i=1}^n \frac{4w_i dnL^2}{l_i})$$

with $c_K = 0$. (6)

If Assumption 2, 3 and 5 are satisfied and α is chosen such that $\gamma := \alpha - \alpha c_0 - \frac{2d}{l}(\frac{\alpha^2 L}{2} + \alpha^2 c_0) > 0$[2]*, then for Algorithm 2 we have*

$$\frac{1}{T}\sum_{s=0}^{S-1}\sum_{k=0}^{K-1} \mathbb{E}||\nabla f(x_k^s)||^2 = \frac{f(x_0^0) - f^*}{T\gamma} \tag{7}$$

where $T = SK$.

Here $\{c_k\}_{k=0}^K$ is used to construct a Lyapunov function $V_k^s = f(x_k^s) + c_k||x_k^s - x_0^s||$ to prove the above theorem as follows

Proof. To simplify the proof, we write g_k^s for $k = 0$ and $k \neq 0$ as one unified form.

$$g_k^s = \sum_{i=1}^n \frac{dw_i}{l_i} P_{k,i}^s((P_{k,i}^s)^T P_{k,i}^s)^{-1}(P_{k,i}^s)^T(\nabla F(x_k^s; \Xi_{k,i}^s) - \nabla F(x_0^s; \Xi_{k,i}^s)) + \frac{d}{l} P_k^s((P_k^s)^T P_k^s)^{-1}(P_k^s)^T \nabla f(x_0^s) \tag{8}$$

In the following, we want to give a bound to a Lyapunov function $V_k^s = f(x_k^s) + c_k||x_k^s - x_0^s||$.

First, by Assumption 2, we have

$$\begin{aligned} f(x_{k+1}^s) &\le f(x_k^s) + \nabla f(x_k^s)^T(x_{k+1}^s - x_k) + \frac{L}{2}||x_{k+1}^s - x_k^s||^2 \\ &\le f(x_k^s) - \alpha \nabla f(x_k^s)^T g_k^s + \frac{\alpha^2 L}{2}||g_k^s||^2 \end{aligned} \tag{9}$$

From Assumption 5, we have $\mathbb{E}[g_k^s|x_k^s] = \nabla f(x_k^s)$. Now we want to bound $\mathbb{E}[||g_k^s||^2|x_k^s]$.

$$\begin{aligned} &\mathbb{E}[||g_k^s||^2|x_k^s] \\ &\le 2\mathbb{E}[||g_k^s - \frac{d}{l}P_k^s((P_k^s)^T P_k^s)^{-1}(P_k^s)^T \nabla f(x_k^s)||^2|x_k^s] \\ &+ 2\mathbb{E}[||\frac{d}{l}P_k^s((P_k^s)^T P_k^s)^{-1}(P_k^s)^T \nabla f(x_k^s)||^2|x_k^s] \qquad (10) \\ &\le 4\mathbb{E}[||\frac{d}{l}P_k^s((P_k^s)^T P_k^s)^{-1}(P_k^s)^T(\nabla f(x_k^s) - \nabla f(x_0^s))||^2|x_k^s] \\ &+ 4\mathbb{E}[||\sum_{i=1}^n \frac{w_i d}{l_i} P_{k,i}^s((P_{k,i}^s)^T P_{k,i}^s)^{-1}(P_{k,i}^s)^T \\ &\quad (\nabla F(x_k^s; \Xi_{k,i}^s) - \nabla F(x_0^s; \Xi_{k,i}^s))||^2|x_k^s] \\ &+ 2\mathbb{E}[||\frac{d}{l}P_k^s((P_k^s)^T P_k^s)^{-1}(P_k^s)^T \nabla f(x_k^s)||^2|x_k^s] \qquad (11) \end{aligned}$$

[2]We can always find an α satisfying the condition by making α small enough. This is because when α is small enough, then c_0 can be less than 1 due to (6). Then γ can be made positive because of its positive α term and negative α^2 term.

$$\leq \frac{4d}{l}||\nabla f(x_k^s) - \nabla f(x_0^s)||^2 + \frac{2d}{l}||\nabla f(x_k^s)||^2$$
$$+ \sum_{i=1}^{n} \frac{4w_i dn}{l_i}||\nabla F(x_k^s;\Xi_{k,i}^s) - \nabla F(x_0^s;\Xi_{k,i}^s)||^2 \qquad (12)$$
$$\leq \frac{4dL^2}{l}||x_k^s - x_0^s||^2 + \frac{2d}{l}||\nabla f(x_k^s)||^2$$
$$+ \sum_{i=1}^{n} \frac{4w_i dnL^2}{l_i}||x_k^s - x_0^s||^2 \qquad (13)$$

where (10) is from $||a+b||^2 \leq 2||a||^2 + 2||b||^2$, (11) is from the definition of g_k^s in (8), (12) is from Assumption 5 and $||\sum_{i=1}^n a_i||^2 \leq n\sum_{i=1}^n ||a_i||^2$, and (13) is from Assumption 2.

Meanwhile, we have

$$\begin{aligned}
&\mathbb{E}[||x_{k+1}^s - x_0^s||^2|x_k^s] \\
&= \mathbb{E}[||x_k^s - x_0^s||^2 + ||x_{k+1}^s - x_k^s||^2 \\
&+ 2(x_{k+1}^s - x_k^s)^T(x_k^s - x_0^s)|x_k^s] \\
&\leq \mathbb{E}[\alpha^2||g_k^s||^2|x_k^s] + ||x_k^s - x_0^s||^2 + \alpha||\nabla f(x_k^s)||^2 \\
&+ \alpha||x_k^s - x_0^s||^2 \qquad (14)
\end{aligned}$$

where (14) is from $2ab \leq ||a||^2 + ||b||^2$ and $\mathbb{E}[x_{k+1}^s - x_k^s|x_k^s] = \alpha\nabla f(x_k^s)$.

Combing (9), (13) and (14), we have

$$\begin{aligned}
&\mathbb{E}[V_{k+1}^s|x_k^s] \\
&\leq f(x_k^s) - \left(\alpha - \alpha c_{k+1} - \frac{2d}{l}(\frac{\alpha^2 L}{2} + \alpha^2 c_{k+1})\right)||\nabla f(x_k^s)||^2 \\
&+ (c_{k+1} + \alpha c_{k+1})||x_k^s - x_0^s||^2 + (\frac{\alpha^2 L}{2} + \alpha^2 c_{k+1}) \\
&(\frac{4dL^2}{l} + \sum_{i=1}^{n}\frac{4w_i dnL^2}{l_i})||x_k^s - x_0^s||^2 \qquad (15)
\end{aligned}$$

Due to (6), we have

$$\begin{aligned}
&\mathbb{E}[V_{k+1}^s|x_k^s] \\
&\leq V_k^s - \left(\alpha - \alpha c_{k+1} - \frac{2d}{l}(\frac{\alpha^2 L}{2} + \alpha^2 c_{k+1})\right)||\nabla f(x_k^s)||^2
\end{aligned}$$

It is easy to see that c_0 is largest among $\{c_0, ..., c_K\}$ from (6). By our choice of α, we have

$$\mathbb{E}[V_{k+1}^s|x_k^s] \leq V_k^s - \gamma||\nabla f(x_k^s)||^2$$

with $\gamma > 0$ defined in Theorem 2, which leads to

$$\sum_{k=0}^{K-1}\mathbb{E}||\nabla f(x_k^s)||^2 \leq \frac{\mathbb{E}[V_0^s - V_K^s]}{\gamma}$$

by telescoping from $k=0$ to $K-1$ within epoch s. Meanwhile, since $c_K = 0$, we have $V_K^s = V_0^{s+1}$. By telescoping all epoches, we have

$$\frac{1}{T}\sum_{s=0}^{S-1}\sum_{k=0}^{K-1}\mathbb{E}||\nabla f(x_k^s)||^2 \leq \frac{\mathbb{E}[V_0^0 - V_K^{S-1}]}{T\gamma} \leq \frac{f(x_0^0) - f^*}{T\gamma}$$

where $T = SK$, $V_0^0 = f(x_0^0)$ and $V_K^{S-1} = f(x_K^{S-1}) \geq f^*$. □

Note that compared to the $O(1/\sqrt{T})$ rate of FL-SSGD, FL-SSVRG has a faster $O(1/T)$ rate without Assumption 1 which is not satisfied for functions like l_2-norm loss. Its rate is also faster than gradient sparsification methods in [6] and [8]. The comparison of sample complexity is determined by T and total sample size in FL. One drawback of FL-SSVRG is that edge devices need extra space to store h_k^s, which makes its expected storage complexity $O(Knl)$ instead of constant complexity of FL-SSGD. However, when Kl is set to be much smaller than d, it still outperforms gradient sparsification methods, whose storage complexity is $O(nd)$.

Now we turn to discussion of how to choose l and l_i to minimize the worst-case uplink communication time of FL-SSVRG based on the bound in Theorem 2. Note that a similar lemma to Lemma 1 holds for Assumption 5.

A. Time Sharing

First, we still suppose that only one device is allowed to transmit at any time of uplink communication and the uplink rate of Device i is r_i. Also suppose that there are at most B bits in each dimension of a vector for accurate representation. Similar to Section III, we can calculate the bound of expected total uplink communication time to reach $\frac{1}{T}\sum_{s=0}^{S-1}\sum_{k=0}^{K-1}\mathbb{E}||\nabla f(x_k^s)||^2 \leq \varepsilon$, which is

$$\sum_{i=1}^{n}\left(\frac{B(l+l_i)}{r_i}(\frac{f(x_0^0) - f^*}{\varepsilon\gamma} - S) + \frac{BlS}{r_i}\right). \qquad (16)$$

Here γ involves parameters of l and l_i shown in Theorem 2. The first term of (16) represents the uplink communication time for $k \neq 0$ and the second term is for $k = 0$. It is complicated to find the best l and l_i due to existence of γ. The only observation is that when l and l_i increase, γ also increases because of (6), which makes the optimality gap (7) of FL-SSVRG smaller. Therefore, FL-SSVRG has a tradeoff between the convergence rate and uplink communication time per iteration ($\frac{B(l+l_i)}{r_i}$ for $k \neq 0$ or $\frac{Bl}{r_i}$ for $k = 0$).

It is not easy to compare (16) with (4) if $l+l_i$ in FL-SSVRG is equal to l_i in FL-SSGD. But we can give an asymptotic result: when ε is small enough, the total uplink communication time of FL-SSVRG will be smaller due to the order of ε in (16) and (4).

B. Channel Sharing

For the case where all the devices share the uplink channel with capacity C, the expected total uplink time to reach $\frac{1}{T}\sum_{s=0}^{S-1}\sum_{k=0}^{K-1}\mathbb{E}||\nabla f(x_k^s)||^2 \leq \varepsilon$ is bounded by$\left(\frac{f(x_0)-f^*}{\varepsilon\gamma} - S\right)\left(\frac{Bl}{r_i} + \mathbb{E}\max_{i\in 1,...,n}\frac{B\tau_{1,i}^0}{r_i}\right) + \frac{BlS}{r_i}$ if we allocate channel capacity r_i to Device i and make $\{P_{k,i}^s\}_{k,s}$ i.i.d.. It is hard to choose l and l_i according to this bound due to complexity of γ. But again, we can see that this bound is smaller in the order of ε compared to FL-SSGD.

V. Simulations

In this section, we will use an example of robust linear regression to test the performance of our algorithms. For a fair comparison, we make sample complexity of the algorithms in the experiments similar to each other.

Robust linear regression studies linear relation of feature-outcome (α, β) from data points with outliers. In our simulations, 100 data points $\{(\alpha_{i,j}, \beta_{i,j}\}_{j=1}^{100}$ of Device i are generated i.i.d. by $\beta_{i,j} = \langle x_0, \alpha_{i,j}\rangle + \varepsilon_{i,j}$, where $\alpha_{i,j} \in \mathbb{R}^{1000}$ is generated from $\mathcal{N}(0, \mathbf{I}_{1000})$ and x_0 is a fixed 1000-dimensional vector. The noise $\varepsilon_{i,j}$ is from a mixed Gaussian $0.9\mathcal{N}(0, 0.2i) + 0.1\mathcal{N}(0, 10000)$. Here we use different noise variances for different devices to reflect statistical heterogeneity of FL in practice [12]. Suppose that there are 10 edge devices in FL. To learn x_0, the objective of the central server is to

$$\min_{\lambda\in\mathbb{R}^{1000}} f(x) = \frac{1}{1000}\sum_{i=1}^{10}\sum_{j=1}^{100}\rho_{Tuc}(\beta_{i,j} - \langle\lambda, \alpha_{i,j}\rangle)^2 \quad (17)$$

where $\rho_{Tuc}(t) = 1 - (1 - (t/100)^2)^3$ when $|t| \le 100$ and $\rho_{Tuc}(t) = 1$ otherwise. ρ_{Tuc} is a Tuckey's bisquare loss function [10] for robustness.

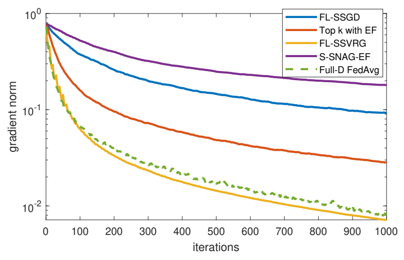
A. Uplink Communication with Time Sharing

In this subsection, we assume that in uplink communication, only one device is allowed to transmit concurrently. The rate of Device i, denoted by r_i, is $100i$ bits per second and we use 1000 bits to represent each dimension of a vector.

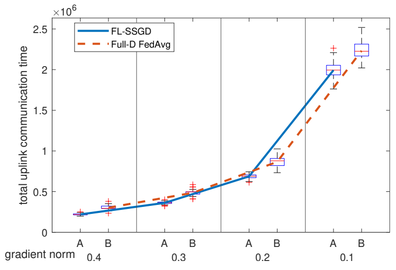
We compare five methods in this experiment, which are FL-SSGD, FL-SSVRG, Top k gradient compressor with error feedback (Top k with EF) [6], S-SNAG-EF [8], and full-dimensional FedAvg [1] as the benchmark. Top k with EF and S-SNAG-EF can represent gradient sparsification methods with two kinds of gradient compressors discussed in Section I-A2. Since we do not know f^*, we cannot obtain the best l_i for FL-SSGD by solving (4). For simplicity and fairness, l_i is chosen to be proportional to r_i and let the device with the largest r_i transmit full-dimensional (i.e., 1000-dimensional) gradients. In Top k with EF and S-SNAG-EF, the number of dimensions sent by each device per iteration is set to be the nearest integer to l_i in FL-SSGD. For FL-SSVRG, we let l_i be the same with FL-SSGD and l equal to 100 in Assumption 5. Subspace matrices in FL-SSGD and FL-SSVRG are generated as discussed in Remark 1.

Figure 2a shows a typical sample path of these five methods in terms of gradient norm versus iterations. We can see that FL-SSVRG and full-dimensional FedAvg have the fastest convergence rate among these methods. It demonstrates the superior performance of FL-SSVRG in convergence rates by only transmitting subspace of gradients.

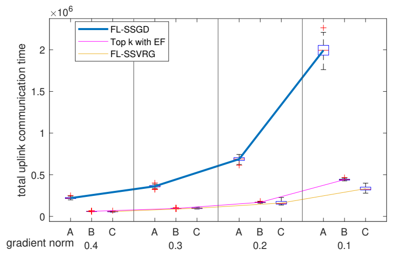
Next we will compare the total uplink communication time to reach a certain gradient norm of (17) in these five methods. To present their statistical results, we run these algorithms for 100 times and show box plots in Figure 2b and Figure 2c for uplink communication time to achieve $0.4, 0.3, 0.2$ and 0.1 gradient norm. We also connect the average of all the runs for each method to reflect its trend. Particularly, Figure 2b compares two methods without extra space requirements,



(a) Convergence rates of five algorithms with regard to gradient norm of f in (17)



(b) Total uplink communication time of two methods without space requirement to achieve a certain gradient norm.



(c) Total uplink communication time of three methods to achieve a certain gradient norm.

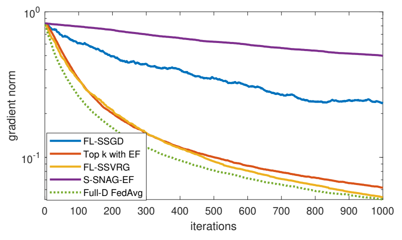
Fig. 2: Methods comparison with time sharing uplink communication.

i.e., FL-SSGD and FedAvg. From the figure, we can see that most sample paths of FL-SSGD can outperform FedAvg regardless of targeted gradient norm and the gap becomes larger as the targeted norm is smaller. FL-SSGD, Top k with EF and FL-SSVRG are compared in Figure 2c, where the latter two methods need extra storage space. Since S-SNAG-EF is too slow to achieve small gradient norms, we neglect it for cleanness of our figure. From the figure, we can see that FL-SSVRG and Top k with EF has comparable uplink communication time to achieve gradient norms of 0.4 and 0.3, but the advantage of FL-SSVRG is more and more obvious as the gradient norm becomes smaller due to the faster convergence rate of FL-SSVRG. Here FL-SSGD has much worse performance than the other two, which means more storage complexity can substantially improve time complexity in this experiment.

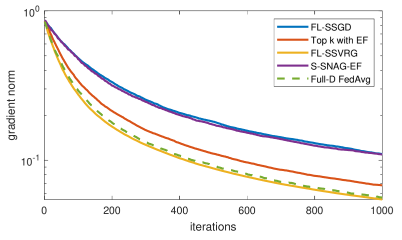
B. Uplink Communication with Channel Sharing

In this subsection, we assume that all the devices share the uplink channel with total capacity $C = 1000$ bits per second.

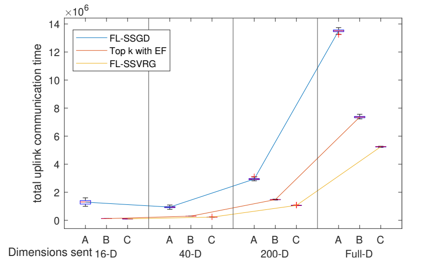
From the discussion in Section III, when using FL-SSGD, we should choose l_i^* by solving (5) with $r_i^* = Cl_i^*/\sum_{i=1}^n l_i^*$,



(a) Convergence rates of four algorithms when 16 dimensions are sent by each device per iteration plus a benchmark.



(b) Convergence rates of four algorithms when 200 dimensions are sent by each device per iteration plus a benchmark.



(c) Total uplink communication time of three methods when each device transmits $16, 40, 200, 1000$ dimensions per iteration.

Fig. 3: Methods comparison with channel sharing uplink communication.

but again f^* is unknown. On the other hands, since $w_1 = ... = w_{10} = 1/1000$ from (17), then all l_i^*'s should be equal from symmetry of (5). As a result, all r_i^*'s should be equal to each other, which means $r_i^* = C/10 = 100$. It remains to decide l_i^*.

In the following, we will test different l_i for FL-SSGD with $r_i = 100, \forall i$. Meanwhile, FL-SSGD will be compared to other methods with the same number of dimensions sent by each device per iteration, including FL-SSVRG, Top k with EF and S-SNAG-EF. In Figure 3a and 3b we plot a sample path of these four algorithms with 16 and 200 dimensions communicated by each device per iteration, respectively, in terms of gradient norm versus iterations. We also add full-dimensional FedAvg as a benchmark for these methods. From Figure 3a and 3b, we can see that except FedAvg, the convergence rates of FL-SSVRG are always the fastest and very close to FedAvg. Meanwhile, FL-SSVRG and Top k with EF are both robust to different numbers of dimensions communicated per iteration, whereas FL-SSGD and S-SNAG-EF become much worse in convergence rates when the number becomes smaller.

Next, we compare total uplink communication time of FL-SSGD, Top k with EF and FL-SSVRG to achieve a gradient norm of 0.1 when the number of dimensions communicated by each device per iteration is $16, 40, 200$ and 1000. S-SNAG-EF is still neglected for cleanness of the figure. Similar to the last subsection, we run these algorithms for 100 times and draw their box plots and connect their averages in Figure 3c. We can see that FL-SSVRG has the least uplink communication time for most sample paths and the gap between FL-SSVRG and the other two methods become larger as the number of dimensions communicated increases. Meanwhile, FL-SSGD with full dimension is equivalent to FedAvg. From Figure 3c, we can see that FedAvg has much worse performance compared to FL-SSGD communicating fewer dimensions per iteration. This shows the necessity of subspace methods for communication time-reduction in FL.

VI. Conclusion

In this paper, we focused on challenges of limited uplink capacity in FL for high-dimensional nonconvex machine learning problems. Inspired by subspace methods, we first proposed a method called FL-SSGD that can reduce communication cost by controlling a parameter in the algorithm without extra storage complexity. To achieve a faster convergence rate with fewer assumptions, we then presented an improved method called FL-SSVRG. Through experiments of robust linear regression problems, we demonstrated advantages of our proposed methods for uplink communication time minimization compared to other benchmark communication-efficient methods.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[3] D. Kozak, S. Becker, A. Doostan, and L. Tenorio, "Stochastic subspace descent," *arXiv preprint arXiv:1904.01145*, 2019.

[4] F. Hanzely, N. Doikov, Y. Nesterov, and P. Richtarik, "Stochastic subspace cubic newton method," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4027–4038.

[5] D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli, "The convergence of sparsified gradient methods," *arXiv preprint arXiv:1809.10505*, 2018.

[6] S. Shi, K. Zhao, Q. Wang, Z. Tang, and X. Chu, "A convergence analysis of distributed sgd with communication-efficient gradient sparsification." in *IJCAI*, 2019, pp. 3411–3417.

[7] H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu, "Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6155–6165.

[8] T. Murata and T. Suzuki, "Accelerated sparsified sgd with error feedback," *arXiv preprint arXiv:1905.12224*, 2019.

[9] Y. Arjevani, Y. Carmon, J. C. Duchi, D. J. Foster, N. Srebro, and B. Woodworth, "Lower bounds for non-convex stochastic optimization," *arXiv preprint arXiv:1912.02365*, 2019.

[10] S. Mei, Y. Bai, A. Montanari *et al.*, "The landscape of empirical risk for nonconvex losses," *Annals of Statistics*, vol. 46, no. 6A, pp. 2747–2774, 2018.

[11] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," in *International conference on machine learning*. PMLR, 2016, pp. 314–323.

[12] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.