Communication-Efficient and Byzantine-Robust Distributed Learning With Error Feedback

Avishek Ghosh, Raj Kumar Maity, Swanand Kadhe, Arya Mazumdar, *Senior Member, IEEE*, and Kannan Ramchandran, *Fellow, IEEE*

***Abstract*—We develop a communication-efficient distributed learning algorithm that is robust against Byzantine worker machines. We propose and analyze a distributed gradient-descent algorithm that performs a simple thresholding based on gradient norms to mitigate Byzantine failures. We show the (statistical) error-rate of our algorithm matches that of Yin *et al.* (2018), which uses more complicated schemes (coordinate-wise median, trimmed mean). Furthermore, for communication efficiency, we consider a generic class of δ-approximate compressors from Karimireddi *et al.* (2019) that encompasses sign-based compressors and top-k sparsification. Our algorithm uses compressed gradients and gradient norms for aggregation and Byzantine removal respectively. We establish the statistical error rate for non-convex smooth loss functions. We show that, in certain range of the compression factor δ, the (order-wise) rate of convergence is not affected by the compression operation. Moreover, we analyze the compressed gradient descent algorithm with error feedback (proposed in Karimireddi *et al.* 2019) in a distributed setting and in the presence of Byzantine worker machines. We show that exploiting error feedback improves the statistical error rate. Finally, we experimentally validate our results and show good performance in convergence for convex (least-square regression) and non-convex (neural network training) problems.**

***Index Terms*—Distributed optimization, communication efficiency, Byzantine resilience, error feedback.**

I. Introduction

In MANY real-world applications, the size of training datasets has grown significantly over the years to the point that it is becoming crucial to implement learning algorithms in a distributed fashion. A commonly used distributed learning framework is data parallelism, in which large-scale datasets are distributed over multiple *worker machines* for parallel processing in order to speed up computation. In other applications such as *Federated Learning* [3], the data sources are inherently distributed since the data are stored locally in users' devices.

In a standard distributed gradient descent framework, a set of worker machines store the data, perform local computations, and communicate gradients to the central machine (e.g., a parameter server). The central machine processes the results from workers to update the model parameters. Such distributed frameworks need to address the following two fundamental challenges. First, the gains due to parallelization are often bottlenecked in practice by heavy communication overheads between workers and the central machine. This is especially the case for large clusters of worker machines or for modern deep learning applications using models with millions of parameters. Moreover, in Federated Learning, communication from a user device to the central server is directly tied to the user's upload bandwidth costs. Second, messages from workers are susceptible to errors due to hardware faults or software bugs, stalled computations, data crashes, and unpredictable communication channels. In scenarios such as Federated Learning, users may as well be malicious and act adversarially. The inherent unpredictable (and potentially adversarial) nature of compute units is typically modeled as *Byzantine failures*. Even if a single worker is Byzantine, it can be fatal to most learning algorithms [4].

Both these challenges, communication efficiency and Byzantine-robustness, have recently attracted significant research attention, albeit mostly separately. In particular, several recent works have proposed various quantization or sparsification techniques to reduce the communication overhead [5], [6], [7], [8], [9], [10], [11], [12]. The goal of these quantization schemes is to compute an unbiased estimate of the gradient with bounded second moment in order to achieve good convergence guarantees. The problem of developing Byzantine-robust distributed algorithms has been considered in [1], [13], [14], [15], [16], [17], [18], [19].

A notable exception to considering communication overhead separately from Byzantine robustness is the recent work of [20]. In this work, a sign-based compression algorithm *signSGD* of [21] is shown to be Byzantine fault-tolerant. The main idea of *signSGD* is to communicate the coordinate-wise signs of the gradient vector to reduce communication and employ a majority vote during the aggregation to mitigate the effect of Byzantine units. However, *signSGD* suffers from

Manuscript received March 1, 2021; revised June 30, 2021 and August 8, 2021; accepted August 10, 2021. Date of publication August 16, 2021; date of current version September 20, 2021. The work of Avishek Ghosh and Kannan Ramchandran was supported in part by NSF under Grant CCF-1527767. The work of Raj Kumar Maity and Arya Mazumdar was supported by NSF under Grant CCF 1642658 and Grant 1618512. The work of Swanand Kadhe was supported in part by National Science Foundation under Grant CCF-1748585 and Grant CNS-1748692. *(Corresponding author: Raj Kumar Maity.)*

Avishek Ghosh was with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA. He is now with HDSI Data Science Institute, University of California at San Diego, La Jolla, CA 92093 USA.

Raj Kumar Maity is with the CICS Department, University of Massachusetts at Amherst, Amherst, MA 01002 USA (e-mail: rajkmaity@cs.umass.edu).

Swanand Kadhe and Kannan Ramchandran are with the EECS Department, University of California at Berkeley, Berkeley, CA 94720 USA.

Arya Mazumdar is with HDSI Data Science Institute, University of California at San Diego, La Jolla, CA 92093 USA.

Digital Object Identifier 10.1109/JSAIT.2021.3105076

two major drawbacks. First, sign-based algorithms do not converge in general [2]. In particular, [2, Sec. 3] presents several convex counter examples where *signSGD* fails to converge even though [20, Th. 2] shows convergence guarantee for non-convex objective under certain assumptions. Second, *signSGD* can handle only a limited class of adversaries, namely *blind multiplicative adversaries* [20]. Such an adversary manipulates the gradients of the worker machines by multiplying it (element-wise) with a vector that can scale and randomize the sign of each coordinate of the gradient. However, the vector must be chosen before observing the gradient (hence 'blind'). In a very recent work [22], authors address the problem of stochastic and compression noise in the presence of Byzantine machines and propose BROADCAST, a variance reduction method with gradient difference compression scheme.

In this work, we develop communication-efficient and robust learning algorithms that overcome both these drawbacks.[1] Specifically, we consider the following distributed learning setup. There are m worker machines, each storing n data points. The data points are generated from some unknown distribution $\mathcal{D}$. The objective is to learn a parametric model that minimizes a population loss function $F : \mathcal{W} \to \mathbb{R}$, where F is defined as an expectation over $\mathcal{D}$, and $\mathcal{W} \subseteq \mathbb{R}^d$ denotes the parameter space. We choose the loss function F to be non-convex. With the rapid rise of the neural networks, the study of *local minima* in non-convex optimization framework has become imperative [23], [24]. For gradient compression at workers, we consider the notion of a δ-approximate compressor from [2] that encompasses sign-based compressors like QSGD [11], ℓ_1-QSGD [2] and top-k sparsification [6]. We assume that $0 \leq \alpha < 1/2$ fraction of the worker machines are Byzantine. In contrast to blind multiplicative adversaries, we consider unrestricted adversaries.

Our key idea is to use a simple threshold (on local gradient norms) based Byzantine resilience scheme instead of robust aggregation methods such as coordinate wise median or trimmed mean of [1]. We mention that similar ideas are used in *gradient clipping*, where gradients with norm more than a threshold is truncated. This is used in applications like training neural nets [25] to handle the issue of exploding gradients, and in differentially private SGD [26], to limit the sensitivity of the gradients.[2]

[1]We compare our algorithm with *signSGD* in Section VIII.

[2]Note that although gradient clipping and norm based thresholding have some similarities, they are not identical. In gradient clipping, although we scale down (clip) the gradients, we retain them. On the other hand, in norm based thresholding, we aim to identify the Byzantine machines and remove them. Note that in our learning framework, we have α fraction of Byzantine workers, and an estimate of α is known to the learning algorithm. When α is very close to 0, our learning algorithm does not trim worker machines, and the effect of all gradients are considered. If we employ gradient clipping in this regime, depending on the threshold used in the clipping operation, some gradients may be scaled back. As a result, the convergence rate will suffer. On the other hand, suppose α is large. In this regime, our algorithm tend to identify and remove the influence of the Byzantine workers, where gradient clipping would scale them down, but retain term in the learning process. This could potentially slow down the learning as the Byzantine machines may send *any arbitrary* updates, which are different for the actual gradient norms and directions. Hence, in both the regimes, the knowledge of α helps our algorithm to handle the Byzantine workers graciously compared to the gradient clipping operation.

Our main result is to show that, for a wide range of compression factor δ, the statistical error rate of our proposed threshold-based scheme is (order-wise) identical to the case of no compression considered in [1]. In fact, our algorithm achieves order-wise optimal error-rate in parameters (α, n, m). Furthermore, to alleviate convergence issues associated with sign-based compressors, we employ the technique of error-feedback from [2]. In this setup, the worker machines store the difference between the actual and compressed gradient and add it back to the next step so that the *correct* direction of the gradient is not forgotten. We show that using error feedback with our threshold based Byzantine resilience scheme not only achieves better statistical error rate but also improves the rate of convergence. We outline our specific contributions in the following.

Our Contributions: We propose a communication-efficient and robust distributed gradient descent (GD) algorithm. The algorithm takes as input the gradients compressed using a δ-approximate compressor along with the norms[3] (of either compressed or uncompressed gradients), and performs a simple thresholding operation based on gradient norms to discard $\beta > \alpha$ fraction of workers with the largest norm values. We establish the statistical error rate of the algorithm for arbitrary smooth population loss functions as a function of the number of worker machines m, the number of data points on each machine n, dimension d, and the compression factor δ. In particular, we show that our algorithm achieves the following statistical error rate[4] for the regime $\delta > 4\beta + 4\alpha - 8\alpha^2 + 4\alpha^3$:

$$\tilde{\mathcal{O}}\left(d^2\left[\frac{\alpha^2}{n} + \frac{1-\delta}{n} + \frac{1}{mn}\right]\right). \tag{1}$$

We first note that when $\delta = 1$ (uncompressed), the error rate is $\tilde{\mathcal{O}}(d^2[\frac{\alpha^2}{n} + \frac{1}{mn}])$, which matches [1]. Notice that we use a simple threshold (on local gradient norms) based Byzantine resilience scheme in contrast with the coordinate wise median or trimmed mean of [1]. We note that for a fixed d and the compression factor δ satisfying $\delta \geq 1-\alpha^2$, the statistical error rate become $\tilde{\mathcal{O}}(\frac{\alpha^2}{n} + \frac{1}{mn})$, which is order-wise identical to the case of no compression [1]. In other words, in this parameter regime, the compression term does not contribute (order-wise) to the statistical error. Moreover, it is shown in [1] that, for strongly-convex loss functions and a fixed d, no algorithm can achieve an error lower than $\tilde{\Omega}(\frac{\alpha^2}{n} + \frac{1}{mn})$, implying that our algorithm is order-wise optimal in terms of the statistical error rate in the parameters (α, n, m).

Furthermore, we strengthen our distributed learning algorithm by using error feedback to correct the direction of the local gradient. We show (both theoretically and via experiments) that using error-feedback with a δ-approximate compressor indeed speeds up the convergence rate and attains better (statistical) error rate. Under the assumption that the gradient norm of the local loss function is upper-bounded by

[3]We can handle any convex norm.

[4]Throughout the paper $\mathcal{O}(\cdot)$ hides multiplicative constants, while $\tilde{\mathcal{O}}(\cdot)$ further hides logarithmic factors.

σ, we obtain the following (statistical) error rate:

$$\tilde{O}\left(d^2\left[\frac{\alpha^2}{n}+\frac{(1-\delta)\sigma^2}{d^2\,\delta}+\frac{1}{mn}\right]\right)$$

provided a similar (δ, α) trade-off.[5] We note that in the no-compression setting ($\delta = 1$), we recover the $\tilde{O}(\frac{\alpha^2}{n} + \frac{1}{mn})$ rate. In experiments (Section VIII), we see that adding error feedback indeed improves the performance of our algorithm.

We experimentally evaluate our algorithm for convex and non-convex losses. For the convex case, we choose the linear regression problem, and for the non-convex case, we train a ReLU activated feed-forward fully connected neural net. We compare our algorithm with the non-Byzantine case and *signSGD* with majority vote, and observe that our algorithm converges faster using the standard MNIST dataset.

A major technical challenge of this paper is to handle compression and the Byzantine behavior of the worker machines simultaneously. We build up on the techniques of [1] to control the Byzantine machines. In particular, using certain distributional assumption on the partial derivative of the loss function and exploiting uniform bounds via careful covering arguments, we show that the local gradient on a non-Byzantine worker machine is close to the gradient of the population loss function.

Note that in some settings, our results may not have an optimal dependence on dimension d. This is due to the norm-based Byzantine removal schemes. Obtaining optimal dependence on d is an interesting future direction.

Organization: We describe the problem formulation in Section II, and give a brief overview of δ-compressors in Section III. Then, we present our proposed algorithm in Section IV. We analyze the algorithm, first, for a *restricted* (as described next) adversarial model in Section V, and in the subsequent section, remove this restriction. In Section V, we restrict our attention to an adversarial model in which Byzantine workers can provide arbitrary values as an input to the compression algorithm, but they correctly implement the same compression scheme as mandated. In Section VI, we remove this restriction on the Byzantine machines. As a consequence, we observe (in Theorem 2) that the modified algorithm works under a stricter assumption, and performs slightly worse than the one in restricted adversary setting. In Section VII, we strengthen our algorithm by including error-feedback at worker machines, and provide statistical guarantees for non-convex smooth loss functions. We show that error-feedback indeed improves the performance of our optimization algorithm in the presence of arbitrary adversaries.

A. Related Work

1) Gradient Compression: The foundation of gradient quantization was laid in [27], [28]. In the work of [9], [10], [11] each co-ordinate of the gradient vector is represented with a small number of bits. Using this, an unbiased estimate of the gradient is computed. In these works, the communication cost is $\Omega(\sqrt{d})$ bits. In [8], a quantization scheme was proposed for distributed mean estimation. The tradeoff between communication and accuracy is studied in [29]. Variance reduction in communication efficient stochastic distributed learning has been studied in [30]. Sparsification techniques are also used instead of quantization to reduce communication cost. Gradient sparsification has beed studied in [5], [6], [7] with provable guarantees. The main idea is to communicate top components of the d-dimensional local gradient to get good estimate of the true global gradient.

[5]See Theorem 3 for details.

2) Byzantine Robust Optimization: In the distributed learning context, a generic framework of one shot median based robust learning has been proposed in [15]. In [16] the issue of Byzantine failure is tackled by grouping the servers in batches and computing the median of batched servers. Later in [1], [17], co-ordinate wise median, trimmed mean and iterative filtering based algorithm have been proposed and optimal statistical error rate is obtained. Also, [31], [32] considers adversaries may steer convergence to bad local minimizers. In this work, we do not assume such adversaries.

Gradient compression and Byzantine robust optimization have simultaneously been addressed in a recent paper [20]. Here, the authors use *signSGD* as compressor and majority voting as robust aggregator. As explained in [2], *signSGD* can run into convergence issues. Also, [20] can handle a restricted class of adversaries that are *multiplicative* (i.e., multiply each coordinate of gradient by arbitrary scalar) and *blind* (i.e., determine how to corrupt the gradient before observing the true gradient). In this paper, for compression, we use a generic δ approximate compressor. Also, we can handle arbitrary Byzantine worker machines.

Very recently, [2] uses error-feedback to remove some of the issues of sign based compression schemes. In this work, we extend the framework to a distributed setting and prove theoretical guarantees in the presence of Byzantine worker machines.

3) Notation: Throughout the paper, we assume $C, C_1, C_2, \ldots, c, c_1, \ldots$ as positive universal constants, the value of which may differ from instance to instance. $[r]$ denotes the set of natural numbers $\{1, 2, \ldots, r\}$. Also, $\|.\|$ denotes the ℓ_2 norm of a vector and the operator norm of a matrix unless otherwise specified.

II. Problem Formulation

In this section, we formally set up the problem. We consider a standard statistical problem of risk minimization. In a distributed setting, suppose we have one central and m worker nodes and the worker nodes communicate to the central node. Each worker node contains n data points. We assume that the mn data points are sampled independently from some unknown distribution $\mathcal{D}$. Also, let $f(w, z)$ be the non-convex loss function of a parameter vector $w \in \mathcal{W} \subseteq \mathbb{R}^d$ corresponding to data point z, where $\mathcal{W}$ is the parameter space. Hence, the population loss function is $F(w) = \mathbb{E}_{z\sim\mathcal{D}}[f(w, z)]$. Our goal is to obtain the following:

$$w^* = \operatorname*{argmin}_{w\in\mathcal{W}} F(w),$$

where we assume $\mathcal{W}$ to be a convex and compact subset of $\mathbb{R}^d$ with diameter D. In other words, we have $\|w_1 - w_2\| \leq D$ for all $w_1, w_2 \in \mathcal{W}$. Each worker node is associated with a local loss defined as $F_i(w) = \frac{1}{n}\sum_{j=1}^{n} f(w, z^{i,j})$, where $z^{i,j}$ denotes the j-th data point in the i-th machine. This is precisely the empirical risk function of the i-th worker node.

We assume a setup where worker i compresses the local gradient and sends to the central machine. The central machine aggregates the compressed gradients, takes a gradient step to update the model and broadcasts the updated model to be used in the subsequent iteration. Furthermore, we assume that α fraction of the total workers nodes are Byzantine, for some $\alpha < 1/2$. Byzantine workers can send any arbitrary values to the central machine. In addition, Byzantine workers may completely know the learning algorithm and are allowed to collude with each other.

Next, we define a few (standard) quantities that will be required in our analysis.

Definition 1 (Sub-Exponential Random Variable): A zero mean random variable Y is called v-sub-exponential if $\mathbb{E}[e^{\lambda Y}] \leq e^{\frac{1}{2}\lambda^2 v^2}$, for all $|\lambda| \leq \frac{1}{v}$.

Definition 2 (Smoothness): A function $h(.)$ is L_F-smooth if $h(w) \leq h(w') + \langle \nabla h(w'), w - w' \rangle + \frac{L_F}{2}\|w - w'\|^2 \;\forall w, w'$.

Definition 3 (Lipschitz): A function $h(.)$ is L-Lipschitz if $\|h(w) - h(w')\| \leq L\|w - w'\| \;\forall w, w'$.

III. Compression At Worker Machines

In this section, we consider a generic class of compressors from [6] and [2] as described in the following.

Definition 4 (δ-Approximate Compressor): An operator $\mathcal{Q}(.) : \mathbb{R}^d \to \mathbb{R}^d$ is defined as δ-approximate compressor on a set $\mathcal{S} \subseteq \mathbb{R}^d$ if, $\forall x \in \mathcal{S}$,

$$\|\mathcal{Q}(x) - x\|^2 \leq (1 - \delta)\|x\|^2,$$

where $\delta \in (0, 1]$ is the compression factor.

Furthermore, a randomized operator $\mathcal{Q}(.)$ is δ-approximate compressor on a set $\mathcal{S} \subseteq \mathbb{R}^d$ if,

$$\mathbb{E}\Big(\|\mathcal{Q}(x) - x\|^2\Big) \leq (1 - \delta)\|x\|^2$$

holds for all $x \in \mathcal{S}$, where the expectation is taken with respect to the randomness of $\mathcal{Q}(.)$. In this paper, for the clarity of exposition, we consider the deterministic form of the compressor (as in Definition 4). However, the results can be easily extended for randomized $\mathcal{Q}(.)$.

Notice that $\delta = 1$ implies $\mathcal{Q}(x) = x$ (no compression). We list a few examples of δ-approximate compressors (including a few from [2]) here:

1) top$_k$ operator, which selects k coordinates with largest absolute value; for $1 \leq k \leq d$, $(\mathcal{Q}(x))_i = (x)_{\pi(i)}$ if $i \leq k$, and 0 otherwise, where π is a permutation of $[d]$ with $(|x|)_{\pi(i)} \geq (|x|)_{\pi(i+1)}$ for $i \in [d-1]$. This is a k/d-approximate compressor.
2) k-PCA that uses top k eigenvectors to approximate a matrix X [9].
3) Quantized SGD (QSGD) [11], where $\mathcal{Q}(x_i) = \|x\| \cdot \mathsf{sign}(x_i) \cdot \xi_i(x)$, where $\mathsf{sign}(x_i)$ is the coordinate-wise sign vector, and $\xi_i(x)$ is defined as following: let $0 \leq l_i \leq s$, be an integer such that $|x_i|/\|x\| \in [l_i/s, (l_i + 1)/s]$. Then, $\xi_i = l_i/s$ with probability $1 - \frac{|x_i|}{c\|x\|\sqrt{d}} + l_i$ and $(l+1)/s$ otherwise. Reference [11] shows that it is a $1 - \min(d/s^2, \sqrt{d}/s)$-approximate compressor.

Algorithm 1 Robust Compressed Gradient Descent

```
1: Input: Step size $\gamma$, Compressor $\mathcal{Q}(.)$, $q > 1$, $\beta < 1$. Also
   define,
   $$\mathcal{C}(x) = \begin{cases} \{\mathcal{Q}(x), \|x\|_q\} & \forall x \in \mathbb{R}^d \quad \text{Option I} \\ \mathcal{Q}(x) & \forall x \in \mathbb{R}^d \quad \text{Option II} \end{cases}$$
2: Initialize: Initial iterate $w_0 \in \mathcal{W}$
3: for $t = 0, 1, \ldots, T-1$ do
4:     Central machine: broadcasts $w_t$
       for $i \in [m]$ do in parallel
5:     $i$-th worker machine:
         • Non-Byzantine:
           – Computes $\nabla F_i(w_t)$; sends $\mathcal{C}(\nabla F_i(w_t))$ to the central machine,
         • Byzantine:
           – Generates $\star$ (arbitrary), and sends $\mathcal{C}(\star)$ to the central machine: Option I,
           – Sends $\star$ to the central machine: Option II,
       end for
6:     Central Machine:
         • Sort the worker machines in a non decreasing order according to
           – Local gradient norm: Option I,
           – Compressed local gradient norm: Option II,
         • Return the indices of the first $1 - \beta$, fraction of elements as $\mathcal{U}_t$,
         • Update model parameter: $w_{t+1} = w_t - \frac{\gamma}{|\mathcal{U}_t|}\sum_{i \in \mathcal{U}_t} \mathcal{Q}(\nabla F_i(w_t))$.
7: end for
```

4) Quantized SGD with ℓ_1 norm [2], $\mathcal{Q}(x) = \frac{\|x\|_1}{d}\mathsf{sign}(x)$, which is $\frac{\|x\|_1^2}{d\|x\|^2}$-approximate compressor. In this paper, we call this compression scheme as ℓ_1-QSGD.

Apart from these examples, several randomized compressors are also discussed in [6]. Also, the *signSGD* compressor, $\mathcal{Q}(x) = \mathsf{sign}(x)$, where $\mathsf{sign}(x)$ is the (coordinate-wise) sign operator, was proposed in [21]. Here the local machines send a d-dimensional vector containing coordinate-wise sign of the gradients.

IV. Robust Compressed Gradient Descent

In this section, we describe a communication-efficient and robust distributed gradient descent algorithm for δ-approximate compressors. The optimization algorithm we use is formally given in Algorithm 1. Note that the algorithm uses a compression scheme $\mathcal{Q}(.)$ to reduce communication cost and a norm based thresholding to remove Byzantine worker nodes. The idea being norm based thresholding is quite intuitive. Note that, if the Byzantine worker machines try to diverge the learning algorithm by increasing the norm of the local gradients; Algorithm 1 can identify them as outliers. Furthermore, when the Byzantine machines behave like inliers, they can not diverge the learning algorithm since they are only a few ($\alpha < 1/2$) in number. It turns out that this simple approach indeed works.

As seen in Algorithm 1, robust compressed gradient descent operates under two different setting, namely *Option I* and *Option II*. Option I and II are analyzed in Sections V and VI respectively. For Option I, we use a δ-approximate compressor along with the norm information. In particular, the worker machines send the pair denoted by $\mathcal{C}(x) = \{\mathcal{Q}(x), \|x\|_q, \}$ where[6] we have $q \geq 1$, to the center machine. $\mathcal{C}(x)$ is comprised of a scalar (norm of x) and a compressed vector $\mathcal{Q}(x)$. For compressors such as QSGD [11] and ℓ_1-QSGD [2], the quantity $\mathcal{Q}(.)$ has the norm information and hence sending the norm separately is not required.

As seen in Option I of Algorithm 1, worker node i compresses the local gradient $\nabla F_i(.)$ sends $\mathcal{C}(\nabla F_i(.))$ to the central machine. Adversary nodes can send arbitrary $\mathcal{C}(\star)$ to the central machine. The central machine aggregates the gradients, takes a gradient step and broadcasts the updated model for next iteration.

For Option I, we restrict to the setting where the Byzantine worker machines can send arbitrary values to the input of the compression algorithm, but they adhere to the compression algorithm. In particular, Byzantine workers can provide arbitrary values, $\star$ to the input of the compression algorithm, $\mathcal{Q}(.)$ but they correctly implement the same compression algorithm, i.e., computes $\mathcal{Q}(\star)$.

We now explain how Algorithm 1 tackles the Byzantine worker machines. The central machine receives the compressed gradients comprising a scalar ($\|x\|_q, q \geq 1$) and a quantized vector ($\mathcal{Q}(x)$) and outputs a set of indices $\mathcal{U}$ with $|\mathcal{U}| = (1-\beta)m$. Here we employ a simple thresholding scheme on the (local) gradient norm.

Note that, if the Byzantine worker machines try to diverge the learning algorithm by increasing the norm of the local gradients; Algorithm 1 can identify them as outliers. Furthermore, when the Byzantine machines behave like inliers, they can not diverge the learning algorithm since $\alpha < 1/2$. In the subsequent sections, we show theoretical justification of this argument.

With Option II, we remove this restriction on Byzantine machines at the cost of slightly weakening the convergence guarantees. This is explained in Section VI. With Option II, the i-th local machine sends $\mathcal{C} = \{\mathcal{Q}(\nabla F_i(w_t)), \|\mathcal{Q}(\nabla F_i(w_t))\|_q\}$ to the central machine, where $q \geq 1$. Effectively, the i-th local machine just sends $\mathcal{Q}(\nabla F_i(w_t))$ since its norm can be computed at the central machine. Byzantine workers just send arbitrary ($\star$) vector instead of compressed local gradient. Note that the Byzantine workers here do not adhere to any compression rule.

The Byzantine resilience scheme with Option II is similar to Option I except the fact that the central machine sorts the worker machines according to the norm of the compressed gradients rather than the norm of the gradients.

V. Distributed Learning With Restricted Adversaries

In this section, we analyze the performance of Algorithm 1 with *Option I*. We restrict to an adversarial model in which Byzantine workers can provide arbitrary values to the input of the compression algorithm, but they adhere to the compression rule. Though this adversarial model is restricted, we argue that it is well-suited for applications wherein compression happens outside of worker machines. For example, Apache MXNet, a deep learning framework designed to be distributed on cloud infrastructures, uses NVIDIA Collective Communication Library (NCCL) that employs gradient compression (see [33]). Also, in a Federated Learning setup the compression can be part of the communication protocol. Furthermore, this can happen when worker machines are divided into groups, and each group is associated with a *compression unit*. As an example, cores in a multi-core processor [34] acting as a group of worker machines with the compression carried out by a separate processor, or servers co-located on a rack [35] acting as a group with the compression carried out by the top-of-the-rack switch.

A. Main Results

We analyze Algorithm 1 (with Option I) and obtain the rate of the convergence under non-convex loss functions. We start with the following assumption.

Assumption 1: For all z, the partial derivative of the loss function $f(., z)$ with respect to the k-th coordinate (denoted as $\partial_k f(., z)$) is L_k Lipschitz with respect to the first argument for each $k \in [d]$, and let $\widehat{L} = \sqrt{\sum_{i=1}^d L_k^2}$. The population loss function $F(.)$ is L_F smooth.

We also make the following assumption on the tail behavior of the partial derivative of the loss function.

Assumption 2 (Sub-Exponential Gradients): For all $k \in [d]$ and z, the quantity $\partial_k f(w, z))$ is v sub-exponential for all $w \in \mathcal{W}$.

The assumption implies that the moments of the partial derivatives are bounded. We like to emphasize that the sub-exponential assumption on gradients is fairly common [1], [14], [36]. For instance, [1, Proposotion 2] gives a concrete example of coordinate-wise sub-exponential gradients in the context of a regression problem. Furthermore, in [17], the gradients are assumed to be sub-gaussian, which is stronger than Assumption 2.

To simplify notation and for the clarity of exposition, we define the following three quantities which will be used throughout the paper.

$$\epsilon_1 = v\sqrt{d}\left(\max\left\{\frac{d}{n}\log\left(1 + 2nD\hat{L}d\right), \times\sqrt{\frac{d}{n}\log\left(1 + 2nD\hat{L}d\right)}\right\}\right) + \frac{1}{n}, \tag{2}$$

$$\epsilon_2 = v\sqrt{d}\left(\max\left\{\frac{d}{(1-\alpha)mn}\log\left(1 + 2(1-\alpha)mnD\hat{L}d\right), \times\sqrt{\frac{d}{(1-\alpha)mn}\log\left(1 + 2(1-\alpha)mnD\hat{L}d\right)}\right\}\right), \tag{3}$$

$$\epsilon = 2\left(1 + \frac{1}{\lambda_0}\right)\left[\left(\frac{1-\alpha}{1-\beta}\right)^2\epsilon_2^2 + \left(\frac{\sqrt{1-\delta} + \alpha + \beta}{1-\beta}\right)^2\epsilon_1^2\right]. \tag{4}$$

[6]Throughout the paper, we use $q = 2$. However, any norm, i.e., $q \geq 1$ can be handled.

where λ_0 is a positive constant. For intuition, one can think of $\epsilon_1 = \tilde{\mathcal{O}}(\frac{d}{\sqrt{n}})$ and $\epsilon_2 = \tilde{\mathcal{O}}(\frac{d}{\sqrt{mn}})$ as small problem dependent quantities. Assuming $\beta = c\alpha$ for a universal constant $c > 1$, we have

$$\epsilon = \tilde{\mathcal{O}}\bigg(d^2\bigg[\frac{\alpha^2}{n} + \frac{1-\delta}{n} + \frac{1}{mn}\bigg]\bigg). \tag{5}$$

Assumption 3 (Size of Parameter Space $\mathcal{W}$): Suppose that $\|\nabla F(w)\| \leq M$ for all $w \in \mathcal{W}$. We assume that $\mathcal{W}$ contains the ℓ_2 ball $\{w{:}\|w - w_0\| \leq c[(2 - \frac{c_0}{2})M + \sqrt{\epsilon}]\frac{F(w_0)-F(w^*)}{\epsilon}\}$, where c_0 is a constant, δ is the compression factor, w_0 is the initial parameter vector and ϵ is defined in equation (4).

We use the above assumption to ensure that the iterates of Algorithm 1 stays in $\mathcal{W}$. We emphasize that this is a standard assumption on the size of $\mathcal{W}$ to control the iterates for non-convex loss function. Note that, similar assumptions have been used in prior works [1, Assumption 5], [17]. We point out that Assumption 3 is used for simplicity and is not a hard requirement. We show (in the proof of Theorem 1) that the iterates of Algorithm 1 stay in a bounded set around the initial iterate w_0. Also, note that the dependence of M in the final statistical rate (implicit, via diameter D) is logarithmic (weak dependence), as will be seen in Theorem 1. Algorithm 1 for T iterations with step size $\gamma = \frac{1}{L_F+\lambda_F}$ yields

We provide the following rate of convergence to a critical point of the (non-convex) population loss function $F(.)$.

Theorem 1: Suppose Assumptions 1, 2 and 3 hold, and $\alpha \leq \beta < 1/2$. For sufficiently small constant c, we choose the step size $\gamma = \frac{c}{L_F}$. Then, running Algorithm 1 for $T = C_3\frac{L_F(F(w_0)-F(w^*))}{\epsilon}$ iterations yields

$$\min_{t=0,\ldots,T} \|\nabla F(w_t)\|^2 \leq C\epsilon,$$

with probability greater than or equal to $1 - \frac{c_1(1-\alpha)md}{(1+n\hat{L}D)^d} - \frac{c_2 d}{(1+(1-\alpha)mn\hat{L}D)^d}$, provided the compression factor satisfies $\delta > \delta_0 + 4\alpha - 9\alpha^2 + 4\alpha^3$, where $\delta_0 = (1 - \frac{(1-\beta)^2}{1+\lambda_0})$ and λ_0 is a (sufficiently small) positive constant.

A few remarks are in order. In the following remarks, we fix the dimension d, and discuss the dependence of ϵ on (α, δ, n, m).

Remark 1 (Rate of Convergence): Algorithm 1 with T iterations yields

$$\min_{t=0,..,T} \|\nabla F(w_t)\|^2 \leq \frac{C_1 L_F(F(w_0) - F(w^*))}{T+1} + C_2\epsilon$$

with high probability. We see that Algorithm 1 converges at a rate of $\mathcal{O}(1/T)$, and finally plateaus at an error floor of ϵ. Note that the rate of convergence is same as [1]. Hence, even with compression, the (order-wise) convergence rate is unaffected.

Remark 2: We observe, from the definition of ϵ that the price for compression is $\tilde{\mathcal{O}}(\frac{1-\delta}{n})$.

Remark 3: Substituting $\delta = 1$ (no compression) in ϵ, we get $\epsilon = \tilde{\mathcal{O}}(\frac{\alpha^2}{n} + \frac{1}{mn})$, which matches the (statistical) rate of [1]. A simple *norm based thresholding* operation is computationally simple and efficient in the high dimensional settings compared to the coordinate wise median and trimmed mean to achieve robustness and obtain the same statistical error and iteration complexity as [1].

Remark 4: When the compression factor δ is large enough, satisfying $\delta \geq 1 - \alpha^2$, we obtain $\epsilon = \tilde{\mathcal{O}}(\frac{\alpha^2}{n} + \frac{1}{mn})$. In this regime, the iteration complexity and the final statistical error of Algorithm 1 is order-wise identical to the setting with no compression [1]. We emphasize here that a reasonable high δ is often observed in practical applications like training of neural nets [2, Fig. 2].

Remark 5 (Optimality): For a distributed mean estimation problem, [1, Observation 1] implies that any algorithm will yield an (statistical) error of $\Omega(\frac{\alpha^2}{n} + \frac{d}{mn})$. Hence, in the regime where $\delta \geq 1 - \alpha^2$, our error-rate is optimal.

Remark 6: For the convergence of Algorithm 1, we require $\delta > \delta_0 + 4\alpha - 9\alpha^2 + 4\alpha^3$, implying that our analysis will not work if δ is very close to 0. Note that a very small δ does not give good accuracy in practical applications [2, Fig. 2]. Also, note that, from the definition of δ_0, we can choose λ_0 sufficiently small at the expense of increasing the multiplicative constant in ϵ by a factor of $1/\lambda_0$. Since the error-rate considers asymptotic in m and n, increasing a constant factor is insignificant. A sufficiently small λ_0 implies $\delta_0 = \mathcal{O}(2\beta)$, and hence we require $\delta > 4\alpha + 2\beta$ (ignoring the higher order dependence).

Remark 7: The requirement $\delta > 4\alpha + 2\beta$ can be seen as a trade-off between the amount of compression and the fraction of adversaries in the system. As α increases, the amount of (tolerable) compression decreases and vice versa.

VI. Distributed Optimization With Arbitrary Adversaries

In this section we remove the assumption of restricted adversary (as in Section V) and make the learning algorithm robust to the adversarial effects of both the computation and compression unit. In particular, here we consider Algorithm 1 with Option II. Hence, the Byzantine machines do not need to adhere to the mandated compression algorithm.

In Option II, the worker machines send $\mathcal{Q}(\nabla F_i(w_t))$ to the center machine. The center machine computes its norm, and discards the top β fraction of the worker machines having largest norm. Note that it is crucial that the center machine computes the norm of $\mathcal{Q}(\nabla F_i(w_t))$, instead of asking the worker machine to send it (similar to Option I). Otherwise, a Byzantine machine having a large $\|\mathcal{Q}(x)\|_q$ can (wrongly) report a small value of $\|\mathcal{Q}(x)\|_q$, gets selected in the trimming phase and influences (or can potentially diverge) the optimization algorithm. Hence, the center needs to compute $\|\mathcal{Q}(x)\|_q$ to remove such issues.

Although this framework is more general in terms of Byzantine attacks, however, in this setting, the statistical error-rate of our proposed algorithm is slightly weaker than that of Theorem 1. Furthermore, the (δ, α) trade-off is stricter compared to Theorem 1.

A. Main Results

We continue to assume that the population loss function $F(.)$ is smooth and non-convex and analyze Algorithm 1 with Option II. We have the following result. For the clarity of exposition, we define the following quantity which will be used in the results of this section:

$$\widetilde{\epsilon} = 2\left(1+\frac{1}{\lambda_0}\right)\left(\left(\frac{(1+\beta)\sqrt{1-\delta}+\alpha+\beta}{1-\beta}\right)^2 \epsilon_1^2 + \left(\frac{1-\alpha}{1-\beta}\right)^2 \epsilon_2^2\right).$$

Comparing $\widetilde{\epsilon}$ with ϵ, we observe that $\widetilde{\epsilon} > \epsilon$. Also, note that,

$$\widetilde{\epsilon} = \tilde{\mathcal{O}}\left(d^2\left[\frac{\alpha^2}{n} + \frac{1-\delta}{n} + \frac{1}{mn}\right]\right), \tag{6}$$

which suggests that $\widetilde{\epsilon}$ and ϵ are order-wise similar. We have the following assumption, which parallels Assumption 3, with ϵ replaced by $\widetilde{\epsilon}$.

Assumption 4 (Size of parameter space $\mathcal{W}$): Suppose that $\|\nabla F(w)\| \leq M$ for all $w \in \mathcal{W}$. We assume that $\mathcal{W}$ contains the ℓ_2 ball $\{w{:}\|w - w_0\| \leq c[(2-\frac{c_0}{2})M + \sqrt{\widetilde{\epsilon}}]\frac{F(w_0)-F(w^*)}{\widetilde{\epsilon}}\}$, where c_0 is a constant, δ is the compression factor and $\widetilde{\epsilon}$ is defined in equation (6).

Theorem 2: Suppose Assumptions 1, 2 and 4 hold, and $\alpha \leq \beta < 1/2$. For sufficiently small constant c, we choose the step size $\gamma = \frac{c}{L_F}$. Then, running Algorithm 1 for $T = C_3 \frac{L_F(F(w_0)-F(w^*))}{\widetilde{\epsilon}}$ iterations yields

$$\min_{t=0,\ldots,T} \|\nabla F(w_t)\|^2 \leq C\widetilde{\epsilon},$$

with probability greater than or equal to $1 - \frac{c_1(1-\alpha)md}{(1+n\hat{L}D)^d} - \frac{c_2 d}{(1+(1-\alpha)mn\hat{L}D)^d}$, provided the compression factor satisfies $\delta > \widetilde{\delta_0} + 4\alpha - 8\alpha^2 + 4\alpha^3$, where $\widetilde{\delta_0} = (1 - \frac{(1-\beta)^2}{(1+\beta)^2(1+\lambda_0)})$ and λ_0 is a (sufficiently small) positive constant.

Remark 8: The above result and their consequences resemble that of Theorem 1. Since $\widetilde{\epsilon} > \epsilon$, the statistical error-rate in Theorem 2 is strictly worse than that of Theorem 1 (although order-wise they are same).

Remark 9: Note that the definition of δ_0 is different than in Theorem 1. For a sufficiently small λ_0, we see $\widetilde{\delta_0} = \mathcal{O}(4\beta)$, which implies we require $\delta > 4\beta + 4\alpha$ for the convergence of Theorem 2. Note that this is a slightly strict requirement compared to Theorem 1. In particular, for a given δ, Algorithm 1 with Option II can tolerate less number of Byzantine machines compared to Option I.

Remark 10: The result in Theorem 2 is applicable for arbitrary adversaries, whereas Theorem 1 relies on the adversary being restrictive. Hence, we can view the limitation of Theorem 2 (such as worse statistical error-rate and stricter (δ, α) trade-off) as a price of accommodating arbitrary adversaries.

VII. Byzantine Robust Distributed Learning With Error Feedback

We now investigate the role of error feedback [2] in distributed learning with Byzantine worker machines. We stick to the formulation of Section I.

In order to address the issues of convergence for sign based algorithms (like *signSGD*), [2] proposes a class of optimization algorithms that uses *error feedback*. In this setting, the worker machine locally stores the error between the actual local gradient and its compressed counterpart. Using this as feedback, the worker machine adds this error term to the compressed gradient in the subsequent iteration. Intuitively, this accounts for correcting the direction of the local gradient. The error-feedback has its roots in some of the classical communication system like "delta-sigma" modulator and adaptive modulator [37].

Algorithm 2 Distributed Compressed Gradient Descent With Error Feedback

```
1: **Input:** Step size $\gamma$, Compressor $\mathcal{Q}(.)$, parameter $\beta (> \alpha)$.
2: **Initialize:** Initial iterate $w_0$, $e_i(0) = 0 \ \forall \ i \in [m]$
3: **for** $t = 0, 1, \ldots, T-1$ **do**
4:     Central machine: sends $w_t$ to all worker
       **for** $i \in [m]$ **do in parallel**
5:     $i$-th non-Byzantine worker machine:
         • computes $p_i(w_t) = \gamma \nabla F_i(w_t) + e_i(t)$
         • sends $\mathcal{Q}(p_i(w_t))$ to the central machine
         • computes $e_i(t+1) = p_i(w_t) - \mathcal{Q}(p_i(w_t))$
6:     Byzantine worker machine:
         • sends $\star$ to the central machine.
7:     At Central machine:
         • sorts the worker machines in non-decreasing order
           according to $\|\mathcal{Q}(p_i(w_t))\|$.
         • returns the indices of the first $1-\beta$ fraction of
           elements as $\mathcal{U}_t$.
         • $w_{t+1} = w_t - \frac{\gamma}{|\mathcal{U}_t|}\sum_{i \in \mathcal{U}_t} \mathcal{Q}(p_i(w_t))$
8: **end for**
```

We analyze the distributed error feedback algorithm in the presence of Byzantine machines. The algorithm is presented in Algorithm 2. We observe that here the central machine sorts the worker machines according to the norm of the compressed local gradients, and ignore the largest β fraction.

Note that, similar to Section VI, we handle arbitrary adversaries. In the subsequent section, we show (both theoretically and experimentally) that the statistical error rate of Algorithm 2 is smaller than Algorithm 1.

A. Main Results

In this section we analyze Algorithm 2 and obtain the rate of the convergence under non-convex smooth loss functions. Throughout the section, we select γ as the step size and assume that Algorithm 2 is run for T iterations. We start with the following assumption.

Assumption 5: For all non-Byzantine worker machine i, the local loss functions $F_i(.)$ satisfy $\|\nabla F_i(x)\|^2 \leq \sigma^2$, where $x \in \{w_j\}_{j=0}^T$, and $\{w_0, \ldots, w_T\}$ are the iterates of Algorithm 2.

Note that several learning problems satisfy the above condition (with high probability). In Appendix in the supplementary material we consider the canonical problem of least squares and obtain an expression of σ^2 with high probability.

Note that since $F_i(.)$ can be written as loss over data points of machine i, we observe that the bounded gradient condition is equivalent to the bounded second moment condition for SGD, and have featured in several previous works, see, e.g., [38], [39]. Here, we are using all the data points and (hence no randomness over the choice of data points) perform gradient descent instead of SGD. Also, note that Assumption 5 is weaker than the bounded second moment condition since we do not require $\|\nabla F_i(x)\|^2$ to be bounded for all x; just when $x \in \{w_j\}_{j=0}^T$.

We also require the following assumption on the size of the parameter space $\mathcal{W}$, which parallels Assumption 3 and 4.

Assumption 6 (Size of Parameter Space $\mathcal{W}$): Suppose that $\|\nabla F(w)\| \le M$ for all $w \in \mathcal{W}$. We assume that $\mathcal{W}$ contains the ℓ_2 ball $\{w : \|w - w_0\| \le \gamma r^* T\}$, where

$$r^* = \epsilon_2 + M + \frac{6\beta\big(1+\sqrt{1-\delta}\big)}{(1-\beta)}\left(\epsilon_1 + M + \sqrt{\frac{3(1-\delta)}{\delta}}\sigma\right) + \sqrt{\frac{12(1-\delta)}{\delta}}\sigma,$$

and (ϵ_1, ϵ_2) are defined in equations (2) and (3) respectively.

Similar to Assumption 3 and 4, we use the above assumption to ensure that the iterates of Algorithm 2 stays in $\mathcal{W}$, and we emphasize that this is a standard assumption to control the iterates for non-convex loss function (see [1], [17]).

To simplify notation and for the clarity of exposition, we define the following quantities which will be used in the main results of this section.

$$\Delta_1 = \frac{9\big(1+\sqrt{1-\delta}\big)^2}{2c(1-\beta)^2}\Big[\alpha^2+\beta^2+(\beta-\alpha)^2\Big] \times \left(\epsilon_1^2 + \frac{3(1-\delta)}{\delta}\sigma^2\right) + \frac{50}{c}\epsilon_2^2, \quad (7)$$

$$\Delta_2 = \frac{L^2}{2}\frac{3(1-\delta)\sigma^2}{c\delta} + \frac{2L\epsilon_2^2}{c} + \left(\frac{1}{2}+L\right)\frac{9\big(1+\sqrt{1-\delta}\big)^2}{c(1-\beta)^2} \times \Big[\alpha^2+\beta^2+(\beta-\alpha)^2\Big]\left(\epsilon_1^2+\frac{3(1-\delta)}{\delta}\sigma^2\right), \quad (8)$$

$$\Delta_3 = \left(\frac{L^2}{100}+25L^2\right)\frac{3(1-\delta)\sigma^2}{c\delta}, \quad (9)$$

where c is a universal constant.

We show the following rate of convergence to a critical point of the population loss function $F(.)$.

Theorem 3: Suppose Assumptions 1, 2, 5 and 6 hold, and $\alpha \le \beta < 1/2$. Then, running Algorithm 1 for T iterations with step size γ yields

$$\min_{t=0,\dots,T} \|\nabla F(w_t)\|^2 \le \frac{F(w_0)-F^*}{c\gamma(T+1)} + \Delta_1 + \gamma\Delta_2 + \gamma^2\Delta_3,$$

with probability greater than or equal to $1 - \frac{c_1(1-\alpha)md}{(1+n\hat{L}D)^d} - \frac{c_2 d}{(1+(1-\alpha)mn\hat{L}D)^d}$, provided the compression factor satisfies $\frac{(1+\sqrt{1-\delta})^2}{(1-\beta)^2}[\alpha^2+\beta^2+(\beta-\alpha)^2] < 0.107$. Here Δ_1, Δ_2 and Δ_3 are defined in equations (7), (8) and (9) respectively.

Remark 11 (Choice of Step Size γ): Substituting $\gamma = \frac{1}{\sqrt{T+1}}$, we obtain

$$\min_{t=0,\dots,T}\|\nabla F(w_t)\|^2 \le \frac{F(w_0)-F^*}{c\sqrt{T+1}} + \Delta_1 + \frac{\Delta_2}{\sqrt{T+1}} + \frac{\Delta_3}{T+1},$$

with high probability. Hence, we observe that the quantity associated with Δ_3 goes down at a considerably faster rate ($\mathcal{O}(1/T)$) than the other terms and hence can be ignored, when T is large.

Remark 12: Note that when no Byzantine worker machines are present, i.e., $\alpha = \beta = 0$, we obtain

$$\Delta_1 = \frac{50}{c}\epsilon_2^2, \quad \Delta_2 = \frac{L^2}{2}\frac{3(1-\delta)\sigma^2}{c\delta} + \frac{2L\epsilon_2^2}{c},$$
$$\Delta_3 = \left(\frac{L^2}{100}+25L^2\right)\frac{3(1-\delta)\sigma^2}{c\delta}.$$

Additionally, if $\delta = \Theta(1)$ (this is quite common in applications like training of neural nets, as mentioned earlier), we obtain $\Delta_2 = C(L^2\sigma^2 + L\epsilon_2^2)$, and $\Delta_3 = C_1L^2$. Substituting $\epsilon_2 = \mathcal{O}(\frac{d}{\sqrt{mn}})$ and for a fixed d, the upper bound in the above theorem is order-wise identical to that of standard SGD in a population loss minimization problem under similar setting [40], [41], [2, Remark 4].

Remark 13 (No Compression Setting): In the setting, where $\delta = 1$ (no compression), we obtain

$$\Delta_1 = \mathcal{O}\left[d^2\left(\frac{\alpha^2}{n}+\frac{1}{mn}\right)\right],$$

and

$$\Delta_2 = \mathcal{O}\left[d^2L\left(\frac{\alpha^2}{n}+\frac{1}{mn}\right)\right],$$

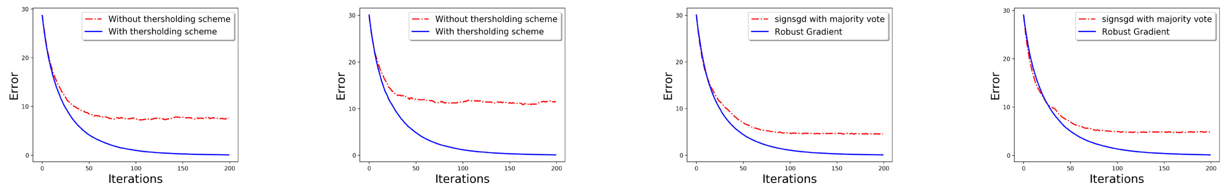
and $\Delta_3 = 0$. The statistical rate (obtained by making T sufficiently large) of the problem is Δ_1, and this rate matches exactly to that of [1]. Hence, we could recover the optimal rate without compression. Furthermore, this rate is optimal in (α, m, n) as shown in [1].

Remark 14 (Comparison With Algorithm 1): In numerical experiments (Section VIII), we compare the performance of Algorithm 2 with the one without error feedback (Algorithm 1). We keep the experiment setup (ex., learning rate, compression) identical for both the algorithms, and compare their performance (see Figure 2). We observe that the convergence of Algorithm 2 with error feedback is faster than Algorithm 1, which is intuitive since error feedback helps in correcting the direction of the local gradient.

VIII. Experiments

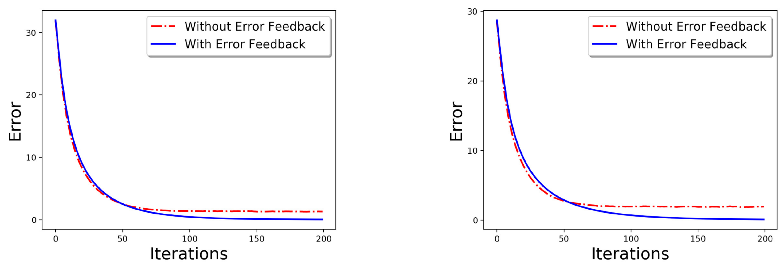
In this section we validate the correctness of our proposed algorithms for linear regression problem and training ReLU network. In all the experiments, we choose the following compression scheme: given any $x \in \mathbb{R}^d$, we report $\mathcal{C}(x) = \{\frac{\|x\|_1}{d}, \mathsf{sign}(x)\}$ where $\mathsf{sign}(x)$ serves as the quantized vector and $\frac{\|x\|_1}{d}$ is the scaling factor. All the reported results are averaged over 20 different runs.

First we consider a least square regression problem $w^* = \text{argmin}_w \|Aw - b\|_2$. For the regression problem we generate matrix $A \in \mathbb{R}^{N\times d}$, vector $w^* \in \mathbb{R}^d$ by sampling each



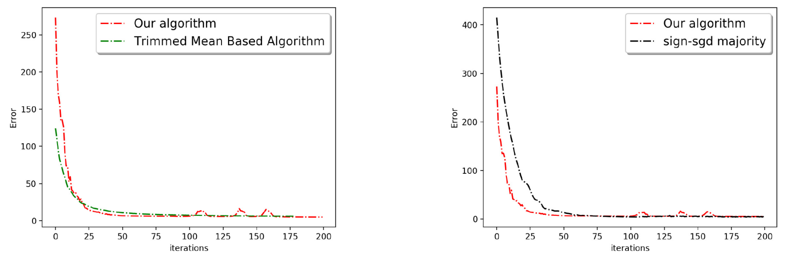
(a) # Byzantine nodes=10 (b) # Byzantine nodes=20 (c) # Byzantine nodes=10 (d) # Byzantine nodes=20

Fig. 1. Comparison of Robust Compressed Gradient Descent with and without thresholding scheme in a regression problem (a,b). The plots show better convergence with thersholding. Comparison of Robust Compressed Gradient Descent with majority vote based *signSGD* [20] in regression Problem. The plots (c,d) show better convergence with thresholding in comparison to the majority vote based robustness of [20].



(a) Number of Byzantine nodes=10 (b) Number of Byzantine nodes=20

Fig. 2. Comparison of norm based thresholding with and without error feedback. The plots show that error feedback based scheme offers better convergence.



(a) (b)

Fig. 3. Training (cross entropy) loss for MNIST image. Comparison with (a) Uncompressed Trimmed mean [1] (b) majority based *signSGD* of [20]. In plot (a) show that Robust Gradient descent matches the convergence of the uncompressed trimmed mean [1]. Plot (b) show a faster convergence compared to the algorithm of [20].

item independently from standard normal distribution and set $b = Aw^*$. Here we choose $N = 4000$ and consider $d = 1000$. We partition the data set equally into $m = 200$ servers. We randomly choose αm $(= 10, 20)$ workers to be Byzantine and apply norm based thresholding operation with parameter βm $(= 12, 22)$ respectively. We simulate the Byzantine workers by adding i.i.d $\mathcal{N}(0, 10 I_d)$ entries to the gradient. In our experiments the gradient is the most pertinent information of the worker server. So we choose to add noise to the gradient to make it a Byzantine worker. However, later on, we consider several kinds of attack models. We choose $\|w_t - w^*\|$ as the error metric for this problem.

A. Effectiveness of Thresholding

We compare Algorithm 1 with compressed gradient descent (with vanilla aggregation). Our method is equipped with Byzantine tolerance steps and the vanilla compressed gradient just computes the average of the compressed gradient sent by the workers. From Figures 1 (a,b) it is evident that the application of norm based thresholding scheme provides better convergence result compared to the compressed gradient method without it.

B. Comparison With signSGD With Majority Vote

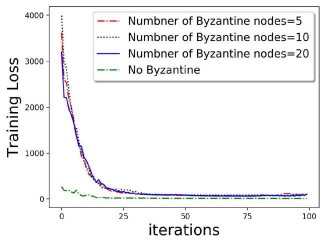
Next, in Figures 1(c,d), we show the comparison of our method with [20] in the regression setup described above. Our method shows a better trend in convergence.

C. Error-Feedback With Thresholding Scheme

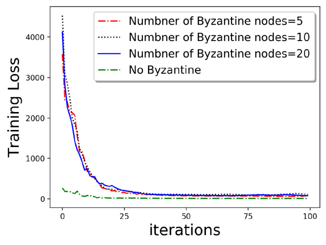
We demonstrate the effectiveness of Byzantine resilience with error-feedback scheme as described in Algorithm 2. We compare our scheme with Algorithm 1 (which does not use error feedback) in Figure 2. It is evident that with error-feedback, better convergence is achieved.

D. Feed-Forward Neural net With ReLU Activation

Next, we show the effectiveness of our method in training a fully connected feed forward neural net. We implement the neural net in pytorch and use the digit recognition dataset MNIST [42]. We partition 60,000 training data into 200 different worker nodes. The neural net is equipped with 1000 node hidden layer with ReLU activation function and we choose *cross-entropy-loss* as the loss function. We simulate the Byzantine workers by adding i.i.d $\mathcal{N}(0, 10 I_d)$ entries to the gradient. In Figure 3 we compare our robust compressed

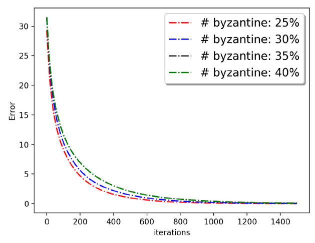


(a) Deterministic shift

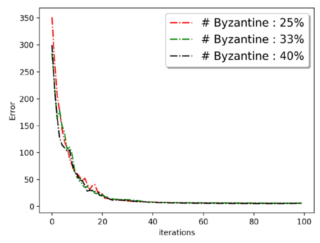


(b) Random Labeling

Fig. 4. Training (cross entropy) loss for MNIST image. Different types of attack (a) labels with deterministic shift (9 − label) (b) random labels. Plots show thresholding scheme with different type of Byzantine attacks achieve similar convergence as ‘no Byzantine’ setup.



(a) Regression Problem



(b) Training loss for ReLU net

Fig. 5. Convergence for (a) regression problem (b) training (cross entropy) loss for MNIST image. Plots show convergence beyond the theoretical bound on the number of Byzantine machine.

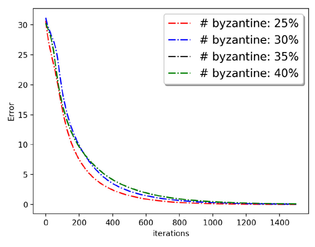
gradient descent scheme with the trimmed mean scheme of [1] and majority vote based *signSGD* scheme of [20]. Compared to the majority vote based scheme, our scheme converges faster. Further, our method shows as good as performance of trimmed mean despite the fact the robust scheme of [1] is an uncompressed scheme and uses a more complicated aggregation rules.

E. Different Types of Attacks

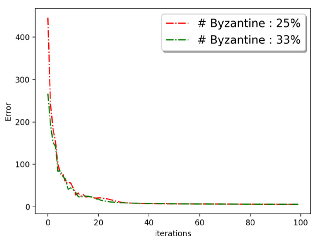
In the previous paragraph we compared our scheme with existing scheme with additive Gaussian noise as a form of Byzantine attack. We also show convergence results with the following type of attacks, which are quite common [1] in neural net training with digit recognition dataset [42]. (a) *Random label:* the Byzantine worker machines randomly replaces the labels of the data, and (b) *Deterministic Shift:* Byzantine workers in a deterministic manner replace the labels y with $9 - y$ (0 becomes 9, 9 becomes 0). In Figure 4 we show the convergence with different numbers of Byzantine workers.

F. Large Number of Byzantine Workers

In Figures 5 and 6, we show the convergence results that holds beyond the theoretical limit (as shown in Theorem 1 and 2) of the number of Byzantine servers in the regression problem and neural net training. In Figure 5, for the regression problem, the Byzantine attack is additive Gaussian noise as described before and our algorithm is robust up to $40\%(\alpha = 0.4)$ of the workers being Byzantine. While training of the feed-forward neural network, we apply a deterministic shift as the Byzantine attack, and the algorithm converges even for $40\%(\alpha = 0.4)$ Byzantine workers.



(a) Regression Problem



(b) Training loss for ReLU net

Fig. 6. Convergence for (a) regression problem (b) training (cross entropy) loss for MNIST image. Plots show convergence with an negative Byzantine attack of $-\epsilon$ times the local gradient with high number of Byzantine machines for $\epsilon = 0.9$.

Another ‘natural’ Byzantine attack would be when a Byzantine worker sends $-\epsilon g$ where $0 \leq \epsilon \leq 1$ and g is the local gradient making the algorithm ‘ascent’ type. We choose $\epsilon = 0.9$ and show convergence for the regression problem for up to 40% Byzantine workers, and for the neural network training for up to 33% Byzantine workers in Figure 6.

IX. Conclusion and Future Work

We address the problem of robust distributed optimization where the worker machines send the compressed gradient to the central machine. We propose a first order optimization algorithm, and consider the setting of restricted as well as arbitrary Byzantine machines. Furthermore, we consider the setup where error feedback is used to accelerate the learning process. We provide theoretical guarantees in all these settings and provide experimental validation under different setup. As an immediate future work, it might also be interesting to study a second order distributed optimization algorithm with compressed gradients and Hessians. In this paper we did not consider a few significant features in Federated Learning: (a) data heterogeneity across users and (b) data privacy of the worker machines. We keep these as our future endeavors.

References

[1] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, “Byzantine-robust distributed learning: Towards optimal statistical rates,” in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.
[2] S. P. Karimireddy, Q. Rebjock, S. U. Stich, and M. Jaggi, “Error feedback fixes SignSGD and other gradient compression schemes,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3252–3261.
[3] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” 2016. [Online]. Available: arXiv:1610.05492.
[4] L. Lamport, R. Shostak, and M. Pease, “The Byzantine generals problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982. [Online]. Available: http://doi.acm.org/10.1145/357172.357176
[5] 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*. Red Hook, NY, USA: Curran, 2018, pp. 5973–5983.
[6] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, “Sparsified SGD with memory,” in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2018, pp. 4447–4458.
[7] N. Ivkin, D. Rothchild, E. Ullah, V. Braverman, I. Stoica, and R. Arora, “Communication-efficient distributed SGD with sketching,” 2019. [Online]. Available: arXiv:1903.04488.

[8] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Proc. 34th Int. Conf. Mach. Learn. Vol. 70*, 2017, pp. 3329–3337. [Online]. Available: JMLR.org

[9] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "ATOMO: Communication-efficient learning via atomic sparsification," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2018, pp. 9850–9861.

[10] W. Wen *et al.*, "TernGrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2017, pp. 1509–1519.

[11] D. Alistarh, D. Grubic, J. Z. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2017, pp. 1709–1720.

[12] V. Gandikota, D. Kane, R. K. Maity, and A. Mazumdar, "vqSGD: Vector quantized stochastic gradient descent," 2019. [Online]. Available: arXiv:1911.07971.

[13] D. Alistarh, D. Grubic, J. Liu, R. Tomioka, and M. Vojnovic, "Communication-efficient stochastic gradient descent, with applications to neural networks," in *Advances in Neural Information Processing Systems 30*. Long Beach, CA, USA: Curran Assoc., Inc., 2017.

[14] L. Su and N. H. Vaidya, "Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms," in *Proc. ACM Symp. Principles Distrib. Comput.*, 2016, pp. 425–434.

[15] J. Feng, H. Xu, and S. Mannor, "Distributed robust learning," 2014. [Online]. Available: arXiv:1409.5937.

[16] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, p. 44, 2017.

[17] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Defending against saddle point attack in Byzantine-robust distributed learning," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 7074–7084. [Online]. Available: http://proceedings.mlr.press/v97/yin19a.html

[18] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Byzantine-tolerant machine learning," 2017. [Online]. Available: arXiv:1703.02757.

[19] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "Robust federated learning in a heterogeneous environment," 2019. [Online]. Available: arXiv:1906.06629.

[20] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, "signSGD with majority vote is communication efficient and fault tolerant," 2018. [Online]. Available: arXiv:1810.05291.

[21] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," 2018. [Online]. Available: arXiv:1802.04434.

[22] H. Zhu and Q. Ling, "BROADCAST: Reducing both stochastic and compression noise to robustify communication-efficient federated learning," 2021. [Online]. Available: arXiv:2104.06685.

[23] D. Soudry and Y. Carmon, "No bad local minima: Data independent training error guarantees for multilayer neural networks," 2016. [Online]. Available: arXiv:1605.08361.

[24] R. Ge, C. Jin, and Y. Zheng, "No spurious local minima in nonconvex low rank problems: A unified geometric analysis," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, Aug. 2017, pp. 1233–1242.

[25] J. Qian, Y. Wu, B. Zhuang, S. Wang, and J. Xiao, "Understanding gradient clipping in incremental gradient methods," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2021, pp. 1504–1512.

[26] X. Chen, S. Z. Wu, and M. Hong, "Understanding Gradient Clipping in Private SGD: A Geometric Perspective," in *Advances in Neural Information Processing Systems*, vol. 33, H. Larochelle, M. Ranzato R. Hadsell, M. F. Balcan, and H. Lin, Eds. Red Hook, NY, USA: Curran Associates, Inc., pp. 13773–13782. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/9ecff5455677b38d19f49ce658ef0608-Paper.pdf

[27] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 1–5.

[28] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 1058–1062.

[29] Y. Zhang, J. Duchi, M. I. Jordan, and M. J. Wainwright, "Information-theoretic lower bounds for distributed statistical estimation with communication constraints," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2013, pp. 2328–2336.

[30] S. Horváth, D. Kovalev, K. Mishchenko, S. Stich, and P. Richtárik, "Stochastic distributed learning with gradient quantization and variance reduction," Apr. 2019. [Online]. Available: arXiv:1904.05115.

[31] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," 2018. [Online]. Available: arXiv:1802.07927.

[32] G. Damaskinos, E. M. El Mhamdi, R. Guerraoui, A. H. A. Guirguis, and S. L. A. Rouault, "AGGREGATHOR: Byzantine machine learning via robust gradient aggregation," in *Proc. Conf. Syst. Mach. Learn. (SysML)*, Stanford, CA, USA, 2019, p. 19. [Online]. Available: http://infoscience.epfl.ch/record/265684

[33] *AWS News Blog*. Accessed: Oct. 8, 2019. [Online]. Available: https://tinyurl.com/yxe4hu4w

[34] K. Lee, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Coded computation for multicore setups," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 2413–2417.

[35] P. Costa, H. Ballani, K. Razavi, and I. Kash, "R2C2: A network stack for rack-scale computers," in *Proc. ACM Conf. Spec. Interest Group Data Commun. (SIGCOMM)*, Aug. 2015, pp. 551–564. [Online]. Available: https://www.microsoft.com/en-us/research/publication/r2c2-a-network-stack-for-rack-scale-computers/

[36] Y. Wu, "Lecture notes for ece598yw: Information-theoretic methods for high-dimensional statistics," Univ. Illinois, Champaign, IL, USA, Rep. ECE598YW, 2016. [Online]. Available: http://www.stat.yale.edu/ yw562/teaching/598/index.html

[37] S. Haykin, *An Introduction to Analog and Digital Communication*. Singapore: Wiley, 1994.

[38] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," 2019. [Online]. Available: arXiv:1910.06378.

[39] P. Mayekar and H. Tyagi, "Limits on gradient compression for stochastic optimization," 2020. [Online]. Available: arXiv:2001.09032.

[40] S. Bubeck, *Convex Optimization: Algorithms and Complexity*. Hanover, MA, USA: Now Publ. Inc., 2015.

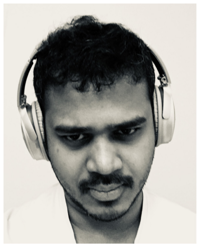
[41] M. Hardt and B. Recht, "Patterns, predictions, and actions: A story about machine learning," 2021. [Online]. Available: arXiv:2102.05242.

[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[43] R. Vershynin, "Introduction to the non-asymptotic analysis of random matrices," 2010. [Online]. Available: arXiv:1011.3027.

[44] M. J. Wainwright, *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*, vol. 48. Cambridge, U.K.: Cambridge Univ. Press, 2019.

Avishek Ghosh received the bachelor's degree from the Department of Electronics and Telecommunication Engineering, Jadavpur University, the master's degree from the Electrical Communication Engineering Department, Indian Institute of Science, Bangalore, India, and the Ph.D. degree from the Electrical Engineering and Computer Sciences Department, UC Berkeley, in 2021, advised by Prof. K. Ramchandran and Prof. A. Guntuboyina. He is an HDSI (Data Science) Postdoctoral Fellow with the University of California San Diego. His research interests are broadly in theoretical machine learning, including federated learning and multiagent reinforcement/bandit learning. In particular, he is interested in theoretically understanding challenges in multiagent systems, and competition/collaboration across agents.



Raj Kumar Maity received the Masters in Science degree from the System Science and Automation Department, Indian Institute of Science (IISc) under the supervision of Prof. S. Bhatnagar. He is currently pursuing the Ph.D. degree with the CICS Department, University of Massachusetts Amherst, Amherst, MA, USA, under the supervision of Prof. A. Mazumdar. He spent a year as a Project Assistant with Stochastic System Lab, CSA Department, IISc. His research interests are distributed optimization, machine learning, differential privacy, error correcting codes, and theory problems in general.

Swanand Kadhe received the Ph.D. degree from Texas A&M University, advised by A. Sprintson. His Ph.D. research was in the area of coding theory, focusing on the aspects of security and availability in large-scale distributed storage systems and private information retrieval. He is a Postdoctoral Researcher with the EECS Department, University of California Berkeley, working with K. Ramchandran. He is interested in designing scalable architectures for distributed systems, focusing on blockchains, and developing efficient algorithms for distributed machine learning.

Arya Mazumdar (Senior Member, IEEE) received the Ph.D. degree from the University of Maryland, College Park, MD, USA, in 2011. He is an Associate Professor of Data Science with the University of California San Diego. From 2015 to 2021, he was an Assistant followed by an Associate Professor with the College of Information and Computer Sciences, University of Massachusetts Amherst. Prior to that, he was a Faculty Member with the University of Minnesota–Twin Cities from 2013 to 2015, and a Postdoctoral Researcher with Massachusetts Institute of Technology from 2011 to 2012. His research interests include coding theory, information theory, statistical learning, and distributed optimization. He is a recipient of multiple awards, including the Distinguished Dissertation Award for his Ph.D. thesis in 2011, the NSF CAREER Award in 2015, the *EURASIP Journal on Advances in Signal Processing* Best Paper Award in 2020, and the IEEE ISIT Jack K. Wolf Student Paper Award in 2010. He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY and as an Area editor for *Foundation and Trends in Communication and Information Theory Series* (Now Publishers).

Kannan Ramchandran (Fellow, IEEE) received the Ph.D. degree from Columbia University in 1993. He has been a Professor of Electrical Engineering and Computer Science with UC Berkeley, since 1999. He was a faculty with UIUC from 1993 to 1999, and with AT&T Bell Labs from 1984 to 1990. He has published extensively in his field, holds over a dozen patents, and has received several awards for his research and teaching including an IEEE Information Theory Society and Communication Society Joint Best Paper award for 2012, an IEEE Communication Society Data Storage Best Paper award in 2010, the two Best Paper awards from the IEEE Signal Processing Society in 1993 and 1999, an Okawa Foundation Prize for outstanding research at Berkeley in 2001, and an Outstanding Teaching Award at Berkeley in 2009, and a Hank Magnuski Scholar award at Illinois in 1998. His research interests are broadly in the area of distributed systems theory and algorithms intersecting the fields of signal processing, communications, coding and information theory, and networking. His current systems focus is on large-scale distributed storage, large-scale collaborative video content delivery, and biological systems, with research challenges including latency, privacy and security, remote synchronization, sparse sampling, and shotgun genome sequencing.