ByGARS: Byzantine SGD with Arbitrary Number of Attackers using Reputation Scores

Jayanth Reddy Regatti [1] **Hao Chen** [1] **Abhishek Gupta** [1]

Abstract

Training distributed machine learning algorithms is prone to Byzantine attacks where the adversarial workers send corrupted model updates to derail the training. Most works focus on the case where less than 0.5 fraction of the workers are adversaries, however, in practical applications it is often possible that the fraction of adversaries is greater than 0.5. In this paper, we propose reputation score based gradient aggregation as a possible solution for this scenario. We introduce a class of novel stochastic gradient descent algorithms, ByGARS (Byzantine Gradient Aggregation using Reputation Scores) that involve computing reputation scores (of workers) to aggregate the model updates using an auxiliary dataset at the server. The computational complexity of ByGARS++ is the same as the usual distributed stochastic gradient descent method with only an additional inner product computation in every iteration. We also demonstrate the effectiveness of the algorithms for non-convex learning problems using MNIST and CIFAR-10 datasets against almost all state-of-the-art Byzantine attacks. We also show that the proposed algorithms are robust to multiple different types of attacks at the same time. See the website for more details, code is available here.

1. Introduction

With increasing data size and model complexity, the preferred method for training machine learning models at scale is to use a distributed training setting. This involves a parameter server that coordinates the training with multiple worker machines by communicating gradients and parameters. Despite the (supposed) speed up in computation due to the distributed setting, it suffers from issues such as straggling workers, while also posing a significant risk to the privacy if the data is collected at a central location. Federated Learning (Konečnỳ et al., 2016; Bonawitz et al., 2019) addresses these issues where the central server only has access to the model parameters/gradients computed by the workers (or independent data owners). However, both the settings are prone to fail in the presence of dishonest workers or non-malicious failed workers (Kairouz et al., 2019).

Thus, there has been a significant interest in devising distributed machine learning schemes in the presence of Byzantine adversaries (Alistarh et al., 2018; Chen et al., 2017; Blanchard et al., 2017; Gupta & Vaidya, 2019). In this setting, a certain fraction of the workers are assumed to be adversarial; instead of sending the actual gradients computed using a randomly sampled mini batch to the server, the adversarial workers send arbitrary or potentially adversarial gradients that could derail the optimization at the server. Several techniques have been proposed to secure the gradient aggregation against adversarial attacks under different settings such as gradient encoding (Chen et al., 2018), asynchronous updates (Damaskinos et al., 2018; Xie et al., 2019b), heterogeneous datasets (Li et al., 2019), decentralized learning (Yang & Bajwa, 2019), (Yang et al., 2019; El-Mhamdi et al., 2019) and Federated Learning (Chang et al., 2019; Portnoy & Hendler, 2020). There has also been some work in developing attack techniques that break existing defenses (Chang et al., 2019; Xie et al., 2019a; Baruch et al., 2019).

One of the main assumptions in past studies about Byzantine attacks in machine learning is that the number of adversarial workers is less than half of the total number of workers. These approaches relied on techniques like majority voting, geometric median, median of means, coordinate wise median, etc., to aggregate gradients at the server. The fundamental reason for this assumption is that the majority based robust statistics approaches require at least more than half of the samples to be correct to give a good estimate. For example, the underlying concept of geometric median has a breakdown point of 0.5 (Chen et al., 2017). In other words,

[1]Electrical and Computer Engineering, The Ohio State University, USA.. Correspondence to: Jayanth Reddy Regatti <regatti.1@osu.edu>.

This work was presented at the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021 (FL-ICML'21). This workshop does not have official proceedings and this paper is non-archival.

Table 1: Summary of various attacks that ByGARS or ByGARS++ is robust to. Extensive simulations suggests that the proposed algorithms are resilient to most of the state-of-the-art attacks with any number of Byzantine adversaries; the checkmarks in the right column indicates that in simulations, we have found either ByGARS or ByGARS++ is able to perform SGD under a wide range of initial conditions. Here, f denotes the fraction of Byzantine adversaries in the system, with $f = 1$ implying that all workers are adversarial.

Type	Attack	Fraction of Adversaries f		
		$f < 0.5$	$f \in [0.5, 1)$	$f = 1$
Omniscient / Collusion	Inner Product Manipulation (Xie et al., 2019a)	√	√	
Omniscient / Collusion	LIE (Baruch et al., 2019)	√	-	
Omniscient / Collusion	OFOM (Chang et al., 2019)	√	√	
Omniscient / Collusion	PAF (Chang et al., 2019)	√	√	
Local / Failure	Sign Flip/Reverse Attack (Blanchard et al., 2017)	√	√	√
Local / Failure	Random Sign Flip Attack	√	√	√
Local / Failure	Gaussian Attack (Blanchard et al., 2017)	√	√	
Local / Failure	Constant Attack (Li et al., 2019)	√	√	
Data Poisoning	Label Flipping	√	√	
Mixed Attacks	Multiple types of attacks	√	√	√

it yields a robust estimator as long as less than half of the data (used for aggregation) is corrupted, and when more than half the data is corrupted, it provably fails.

The assumption that less than half of the workers are adversarial might not be practical. A more practical and challenging problem is to ensure convergence even in the presence of an arbitrary number of adversaries. Some prior works that address this case assume access to some auxiliary (and clean) data (Xie et al., 2018; Jin et al., 2019; Cao & Lai, 2019; Xie et al., 2019b), which is used to identify adversarial workers and the gradients obtained from such workers are discarded at the server. These methods, however require the number of adversaries (or an upper bound) which is crucial to filtering out the adversarial workers but this information may not be available in practical scenarios. In contrast, we propose a reputation score based aggregation, thus obviating the need to know the number of adversaries which make the proposed approach more applicable to practical scenarios. In the proposed approach, we compute the *reputation score* of each worker that is used for gradient aggregation using the auxiliary dataset at the server. The *reputation score* of a worker signifies how relevant the corresponding gradient direction is to the optimization problem, with the intuition being that workers with positive reputation score as being helpful towards the optimization and reputation scores with zero or negative values as being irrelevant or adversarial towards the optimization respectively. As we will see, we achieve robustness to any number of adversaries using the proposed approach.

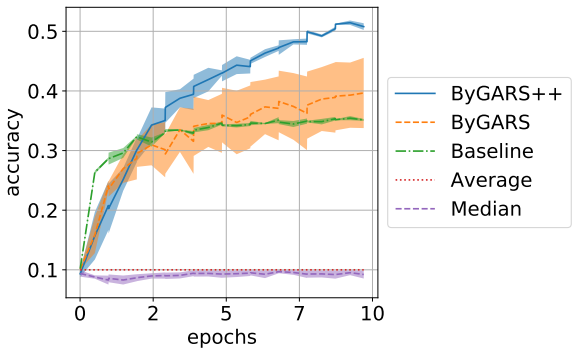

Figure 1: Comparison of the top-1 accuracy of ByGARS and ByGARS++ using CIFAR-10 dataset with one benign worker and seven attackers using different attack strategies. The seven attackers include one Gaussian adversary, two Sign flip adversaries, one random sign flip adversary, two label flip adversaries, and one constant value adversary. See Section 4 for more details.

The key insight that allows us to train under any number of adversaries is that (a) the gradients are being computed for a specific objective function, and (b) the correct gradients would make a small angle with respect to the gradient computed using the auxiliary data with high probability (when the current parameters are far from the optimal ones). To see this, note that geometric median is a robust aggregator under any objective function. Thus, it necessarily requires a more stringent assumption on the number of adversaries to compute a reliable estimate. We explicitly use an unbiased estimate of the objective function (through the auxiliary dataset, and expectation is taken with respect to the draw of the dataset) in our algorithm, which eases the computation and allows us to not use general purpose robust aggregator like geometric median. Further, since the auxiliary dataset has (roughly) the same distribution as the original dataset,

we expect that two stochastic gradients (computed on small mini-batches of the two datasets) would make a small angle with each other with high probability. On the other hand, a random vector will be almost orthogonal to the correct stochastic gradients due to the intrinsic properties of random high dimensional vectors. This intuition allows us to compute the reputation score for each worker reliably early on in the training process. Indeed, we show that our algorithm enjoys convergence under reasonable assumptions such as strongly convex objective function and stationary multiplicative adversaries. Empirical evidence suggests that our algorithm is robust to a large class of Byzantine attacks (summarized in Table 1), not just the ones with multiplicative noise adversaries (sign flip, random sign flip attacks).

Our Contributions: We summarize our main contributions as follows

1. We propose a novel reputation score based gradient aggregation method for distributed machine learning with learnable reputation scores. The idea of reputation scores may be of independent interest for general purpose distributed/Federated learning applications
2. We show that our algorithm is Byzantine tolerant (in the sense of (Xie et al., 2019a)) to an arbitrary number of attackers. Previous works that depend on filtering out the adversaries (for the case of arbitrary number of adversaries) required an estimate on the number of adversaries (often not available in practice). Our proposed algorithms, on the other hand, do not require such assumptions.
3. We use two time-scale stochastic approximation theory to establish the convergence of the proposed algorithm under reasonable assumptions (with strongly convex loss function)
4. Empirical evidence on strongly convex and non-convex objectives suggests that our proposed algorithms are robust to almost all state-of-the-art Byzantine attacks. We also show that our algorithms can defend *mixed attacks* where multiple different attacks are performed at the same time (see Fig 1 and Section 4.5). To the best of our knowledge, we are the first work to demonstrate such ability

2. Problem setup

We consider distributed machine learning with a *parameter server - worker* setup. The parameter server maintains the model parameters, and updates the parameters with gradients received from the workers. We denote the model parameters by $\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^d$, and the number of workers by m. We assume that each worker j has access to dataset, $D_j := \{x_i^j, y_i^j\}_{i=1}^{n_j} \sim \mathcal{D}$, where $N = \sum_j n_j$ is the total number of data points. In the traditional distributed machine learning scenario, the server assigns the data partitions to the workers uniformly at random. In the Federated Learning scenario, this translates to each worker having its own dataset, which is not shared with anyone (not even the server). Given a loss function $f(\cdot, x, y) : \mathbb{R}^d \to \mathbb{R}$, $x, y \sim \mathcal{D}$, the objective is to minimize the population loss $F : \mathbb{R}^d \to \mathbb{R}$

$$\mathbf{w}_* = \arg\min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}) := \mathbb{E}_{x,y\sim\mathcal{D}}[f(\mathbf{w}, x, y)] \quad (1)$$

We denote the true gradient of the population loss at $\mathbf{w}_t$ by $\nabla F(\mathbf{w}_t)$. A good worker samples a subset of the data $\mathcal{D}_{j,t} \subset \mathcal{D}_j$, and computes a stochastic gradient $\tilde{h}_{t,j} := \frac{1}{|\mathcal{D}_{j,t}|}\sum_{x,y\in\mathcal{D}_{j,t}} \nabla f(\mathbf{w}_t, x, y)$. The good workers communicate the stochastic gradient $h_{t,j} := \tilde{h}_{t,j}$ to the server, where as adversarial workers send an arbitrary vector drawn from some distribution (either by computing the stochastic gradient on its subset of data and modifying the gradient adversarially, or by sending an arbitrary random vector). We assume that this attack distribution remains fixed for the adversary throughout the training. We denote the set of gradients received by the server as $H_t^T = [h_{t,1}, \cdots, h_{t,m}] \in \mathbb{R}^{d\times m}$. Note that we assume a synchronous setting here, i.e. all the workers communicate the gradients at the same time to the server. We assume that the server has access to an auxiliary dataset $D_{aux} := \{x_i, y_i\}_{i=1}^n \sim \mathcal{D}$. The server can sample a subset $\xi_{aux,t}$ of the auxiliary dataset and compute auxiliary loss $L_t(\mathbf{w}) = \frac{1}{|\xi_{aux,t}|}\sum_{(x,y)\in\xi_{aux,t}} f(\mathbf{w}, x, y)$, such that $\mathbb{E}[\nabla L_t(\mathbf{w}_t)] = \nabla F(\mathbf{w}_t)$ when the expectation is taken with respect to the draw of the dataset.

3. Algorithm

In this section, we first motivate the importance of using a reputation score for gradient aggregation and introduce the ByGARS class of algorithms to compute the reputation scores and aggregate gradients. Towards the end of the section, we will provide more understanding of the reputation scores based on the proposed algorithms.

In an ideal environment, where all the workers are benign, the gradient aggregation function simply averages the received stochastic gradients and uses the averaged gradient to update the parameters. If the batch sizes used by the workers are different, then a weighted averaging of the stochastic gradients is performed. However, our problem setup is far from ideal, it involves an arbitrary number of workers that act as Byzantine adversaries (potentially all of them can be adversarial).

To compute a meaningful estimate of the gradient, the server maintains a reputation score $q_{t,j}$ for each worker j. Since the adversary can be of any type, the reputation scores can

Algorithm 1 ByGARS: Byzantine Gradient Aggregation using Reputation Scores

$\mathbf{w}_0$ initialized randomly and sent to workers
$\mathbf{q}_0 = \mathbf{0}$
for $t = 1, \cdots, T$ **do**
  receive $H_t^T = [h_{t,1}, \cdots, h_{t,m}]$ from workers
  $\mathbf{q}_{t+1}^0 = \mathbf{q}_t$
  for $i = 1, \cdots, k$ **do**
    $\widehat{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_{t+1}^{i-1}$
    $\mathbf{q}_{t+1}^i \leftarrow \mathbf{q}_{t+1}^{i-1} + \alpha_t \gamma_t H_t \nabla L_t(\widehat{\mathbf{w}}_{t+1})$
  end for
  $\mathbf{q}_{t+1} = \mathbf{q}_{t+1}^k$
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_{t+1}$
  Send $\mathbf{w}_{t+1}$ to workers
end for
Return $\mathbf{w}_{T+1}$

take any real value. The intuition of the reputation score is that, the aggregated gradient is a descent direction for the objective. However, the population loss is not known and is only available through the empirical loss using data at the workers (whom we do not trust). Therefore, we use the auxiliary dataset as a proxy to test whether the aggregated gradient is a descent direction of the original population loss. We will refer to this as the *meta objective* or *auxiliary loss*, and the resultant updates to the reputation score as the *meta updates*.

Suppose, at time t, the reputation score vector is $\mathbf{q}_t = [q_{t,1}, \ldots, q_{t,m}]^T$ and the received gradients are H_t, then the weighted aggregation of the gradients with the reputation score is $H_t^T \mathbf{q}_t = \sum_{i=1}^m q_{t,i} h_{t,i}$. Consider the non-adversarial case where all workers correctly send the computed gradients. Then $q_{t,i} = 1/m$, and $-H_t^T \mathbf{q}_t$ is a descent direction (and an unbiased estimate of the gradient). Consider the case where we know the which workers are adversarial, simply making the reputation scores $q_{t,i}$ for those workers equal to 0 is enough to defend against the attacks.

The problem now is to compute a good reputation score for the workers using only the gradients sent to the server, such that the aggregated gradient is a descent direction for the objective. Making use of our two key assumptions – availability of an auxiliary dataset and the stationary behavior of the workers, we propose the ByGARS (Byzantine Gradient Aggregation using Reputation Scores) class of algorithms.

3.1. ByGARS

We start with an initial reputation score of $\mathbf{q}_0 = \mathbf{0} \in \mathbb{R}^m$, and iteratively improve the estimate of the reputation score. At step t, we perform a *pseudo update* to $\mathbf{w}_t$ (γ_t is a step size parameter) as

$$\hat{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t \tag{2}$$

If $\mathbf{q}_t$ is a good reputation score and γ_t is sufficiently small, then $-H_t^T \mathbf{q}_t$ is a descent direction and thus $F(\hat{\mathbf{w}}_{t+1})$ must be lower in value than $F(\mathbf{w}_t)$ or other points in its neighborhood. However, we neither have access to the true function F nor the data from the workers. Instead, we have a small auxiliary dataset that is drawn from the same distribution as the data at the workers. This auxiliary dataset allows us to construct the loss function $L_t(\cdot)$ (see Section 2), and we can solve the following optimization problem to compute a better reputation score $\mathbf{q}_{t+1}^*$:

$$\mathbf{q}_{t+1} = \arg\min_{\mathbf{q} \in \mathbb{R}^m} L_t(\mathbf{w}_t - \gamma_t H_t^T \mathbf{q}) \tag{3}$$

Using the current estimate $\mathbf{q}_t$, we use an iterative update rule. We compute the loss on a random mini-batch of the auxiliary dataset D_{aux} using $\hat{\mathbf{w}}_t$, which is denoted as $L_t(\hat{\mathbf{w}}_t) = L_t(\mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t)$, and henceforth referred to as auxiliary loss. The objective of the *meta update* is to minimize this loss with respect to $\mathbf{q}$, given $H_t, \mathbf{w}_t$; in other words, find a descent direction on the auxiliary loss at the current iterates. We do this by performing a first order update to $\mathbf{q}_t$ by computing the gradient of the auxiliary loss evaluated at $\hat{\mathbf{w}}_t$ wrt $\mathbf{q}_t$. We refer to this gradient as the auxiliary gradient denoted by $\nabla L_t(\hat{\mathbf{w}}_t)$. The gradient computation and *meta update* to $\mathbf{q}_t$ (α_t is a step size parameter) is given by

$$\begin{aligned} \mathbf{q}_t \leftarrow\ & \mathbf{q}_t - \alpha_t \frac{d}{d\mathbf{q}_t} L_t(\mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t) \\ &= \mathbf{q}_t - \alpha_t(-\gamma_t H_t) \nabla L_t(\mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t) \\ &= \mathbf{q}_t + \alpha_t \gamma_t H_t \nabla L_t(\hat{\mathbf{w}}_t) \end{aligned} \tag{4}$$

The updated reputation score is used to find the updated gradient aggregation $H_t^T \mathbf{q}_t$. The algorithm proceeds by successively applying the *pseudo update* (eq 2) and *meta update* to $\mathbf{q}_t$ (eq 4) for k iterations (or until a stopping criteria is reached, such as sufficient descent) to obtain $\mathbf{q}_{t+1}$ before finally performing an *actual update* to $\mathbf{w}_t$ as

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_{t+1} \tag{5}$$

This is summarized in Algorithm 1 (please note the change in notation, eg. superscript i, due to the meta updates).

3.2. ByGARS++: Faster ByGARS

Algorithm 1 has an additional computational overhead due to multiple parameter updates and multiple gradient computations to update the reputation score in the meta updates.

Algorithm 2 ByGARS++: Faster algorithm to compute Reputation Scores

1: $\mathbf{w}_0$ initialized randomly and sent to workers
2: $\mathbf{q}_0 = \mathbf{0}$
3: **for** $t = 1, \cdots, T$ **do**
4: Receive $H_t^T = [h_{t,1}, \cdots, h_{t,m}]$ from workers
5: Compute $\nabla L_t(\mathbf{w}_t)$ using a subset of the auxiliary data
6: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t$
7: Send $\mathbf{w}_{t+1}$ to workers
8: $\mathbf{q}_{t+1} \leftarrow (1 - \alpha_t)\mathbf{q}_t + \alpha_t H_t \nabla L_t(\mathbf{w}_t)$
9: **end for**
10: **Return** $\mathbf{w}_{T+1}$

This increased computation at the server keeps the workers idle and waiting, thus negating the computational speed-up achieved from distributed learning. In order to overcome this limitation, and driven by the motivation of ByGARS, we propose a variant (ByGARS++) which is computationally cheaper yet efficient and use it to prove theoretical convergence guarantees (see main paper here). We also observe that the reputation scores computed with ByGARS++ have completely different properties than those computed with ByGARS (see here).

We propose Algorithm 2 (ByGARS++), in which we avoid computing multiple *pseudo updates* $\hat{\mathbf{w}}_t$ used for performing *meta updates*, by simulataneously updating $\mathbf{w}_t, \mathbf{q}_t$ as given by eq (6). Note that we perform an update to $\mathbf{q}_t$ using the auxiliary gradients evaluated at $\mathbf{w}_t$ (and not at $\hat{\mathbf{w}}_t$), and we use stochastic approximation for the update.

$$\begin{aligned} \mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t, \\ \mathbf{q}_{t+1} &\leftarrow (1 - \alpha_t)\mathbf{q}_t + \alpha_t H_t \nabla L_t(\mathbf{w}_t) \end{aligned} \tag{6}$$

In this case, the reputation score of each worker is updated using only the inner product between the gradient sent by the worker, and the auxiliary gradient, both evaluated at $\mathbf{w}_t$. The only additional computation as compared to traditional distributed SGD is the udpate of $\mathbf{q}_t$ which takes $\mathcal{O}(md)$ time. However, the server can update $\mathbf{q}_t$ when the workers are computing the gradients for next time step (line 7, 8 of Algorithm 2), therefore ByGARS++ has the same computational complexity as traditional distributed SGD.

3.3. Reputation Scores

As we see from the above algorithms, we compute the reputation score of each worker by taking an inner product between the stochastic gradient (sent by the worker) and the stochastic gradient computed on the auxiliary data.

If a worker consistently sends gradients that are not in the descent direction (of the optimization at the respective iterates), then the reputation score is either zero or accumulates negative values (since the inner product between the worker gradient and auxiliary gradient is either negative or close to zero in expectation). Therefore, by multiplying the received worker gradient by the reputation score, either its impact on the aggregated gradient is reduced (when the reputation score is close to zero) or recover the actual direction of descent (when the reputation score is negative and the worker sends gradients that make an obtuse angle with the auxiliary/true gradient in expectation).

When the parameters are far away from the optima, the inner product between the benign gradients and the auxiliary gradients are positive and higher in magnitude, and therefore contribute heavily towards the reputation scores. Whereas when we are closer to the optima, the inner product value is random (Chee & Toulis, 2018) (due to the directions of the stochastic gradients being random), and hence does not contribute to the reputation score. If unheeded, this phenomenon can destroy the reputation of good workers/boosts that of adversaries, therefore we employ a decaying learning rate schedule for both γ_t and α_t. Thus, by the time the parameters are close enough to the optima or a flat region (in non-convex settings), the learning rates would have decayed significantly. This enables the reputation score to accumulate over time and converge; therefore, the score is robust to the noisy inner products near the optima. As we will see in Section **??**, the decaying learning rate is required for the analysis of the algorithm under the two-timescale stochastic approximation theory.

It is important to note that the algorithms rely on the availability of the auxiliary dataset at the worker. It is a reasonable assumption to make since in practical scenarios, it is not difficult to procure a small amount of clean auxiliary data without violating the privacy of the worker's data. This data can be taken from publicly available datasets (that match the distribution of the data available at the workers), from prior data leaks (that is now publicly available), or data given voluntarily by workers. We provide more analysis on the affect of auxiliary dataset size on the performance of our algorithms in Section 4.5 and observe that a very small auxiliary dataset size is sufficient.

4. Simulations

In this section, we discuss the experimental setup used to evaluate the proposed algorithms.

4.1. Dataset and models

We present the results of our algorithms on MNIST (LeCun et al., 2010), CIFAR-10 (Krizhevsky et al., 2009) for multi-class classification using supervised learning. We used a LeNet architecture (LeCun et al., 1998) for MNIST, and

a two convolutional layer CNN for CIFAR-10. For each dataset, we set aside a small auxiliary dataset of size 250 (sampled randomly from the training data) at the server, and the remaining training data is distributed uniformly to the workers. More details on the hyperparameters, are provided in Appendix **??**.

4.2. Attack mechanisms

The summary of the attacks used in this work is given in Table 1. The attacks were grouped broadly into (i) Omniscient / Collusion attacks, (ii) Local attacks / System failures, and (iii) Data Poisoning attacks. We further propose a *mixed attack*, where we combine multiple Local attacks and Data Poisoning attacks.

In the **Omniscient / Collusion** attacks, the adversaries have complete information about all other workers including the benign ones, or only about the other adversaries. In LIE attack (Baruch et al., 2019), the adversary adds well crafted perturbations to the empirical mean of the benign gradients that is sufficient to avoid convergence. In OFOM, PAF (Chang et al., 2019) a large arbitrary vector is added to the empirical mean of the benign gradients and sent to the server. In Inner Product Manipulation, Xie et al. (Xie et al., 2019a) multiply the empirical mean of benign gradients with a negative value.

In **Local** attacks, the attacker doesn't have any information about the other workers. Instead, the worker either sends an arbitrary gradient to the server or uses the gradient it computed. Examples of the first case include a Gaussian Attack (a vector drawn from a Gaussian distribution of mean 0, covariance 200) (Blanchard et al., 2017), or Constant attack (a vector of all 1s multiplied by an arbitrary scalar, say 100) (Li et al., 2019). In the second case, the attacker computes a gradient on locally computed data, and sends a negatively scaled value of the gradient (this is termed as reverse attack, sign flipping attack in various works) (Blanchard et al., 2017; Li et al., 2019; Bernstein et al., 2018; Jin et al., 2019). Hardware/communication failures that corrupt the gradients (unintentionally) by flipping the sign bit can also be included under these attacks. In addition to the sign flipping attack, we propose a *random* sign flip attack, where at each iteration the adversary draws a real number from a fixed distribution (optionally scales it with a large constant), multiplies with the local gradient and sends to the server. Note that the mean of the distribution can take negative as well as positive values. In our simulations we used a Gaussian distribution with mean randomly picked around -2 and variance 1.

In **Data Poisoning** attacks, the underlying data used to compute the gradients is poisoned so that the model outputs the attacker chosen targets during inference (Baruch et al., 2019). There are several varieties of data poisoning (also called backdooring) attacks, but we only consider label flipping attack in this work. Under the label flipping attack, for example in MNIST dataset, the attacker maps the labels as $l \to (9 - l)$ for $l \in \{0, \cdots, 9\}$, and uses these labels for computing the gradients. Note that, label flipping attack is also a Local attack.

In addition to these attacks, we propose a *Mixed Attack* where there can be multiple Local attacks and Data Poisoning attacks. This is motivated by the need to develop algorithms that are robust to several different types of attacks at the same time. We summarize all these attacks in Table 1.

4.3. Baselines

The generalization performance of SGD improves with the size of the dataset (Hardt et al., 2015). Since existing techniques for arbitrary number of Byzantines filter out the gradients that are sent by the adversarial workers, (in a true Federated Setting) the generalization performance of those techniques is limited by the number of benign workers (and the number of byzantine attackers perceived by the algorithm) and the amount of data available with them. In the case where all workers are adversaries, the only available truthful data is the auxiliary dataset. Hence, we consider plain averaging of the gradients available at these benign workers (and auxiliary gradient) as the *Baseline*. Note that this is the **best** any Byzantine resilient algorithm that relies on filtering out adversarial gradients can do. Primarily for this reason, along with other implementation specific details required for other works that consider an arbitrary number of adversaries (Zeno requires trim parameter b that needs knowledge of the number of adversaries), we chose to compare our algorithm with *Baseline* (described above) as the gold standard. In addition to this, for illustration purposes, we also consider plain averaging of all gradients (no defense) denoted by *Average*, and coordinate-wise median (Yin et al., 2018) denoted by *Median* in our empirical

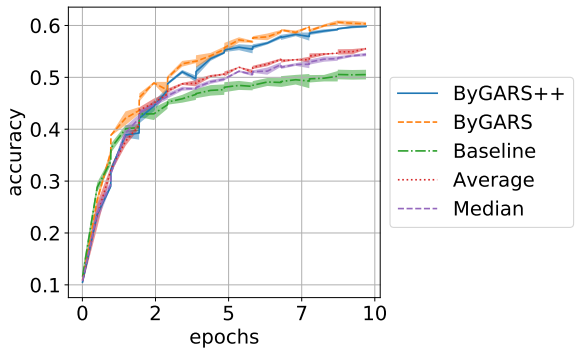

Figure 2: This figure shows the top-1 accuracy of models trained on CIFAR-10 dataset under *No Attack*

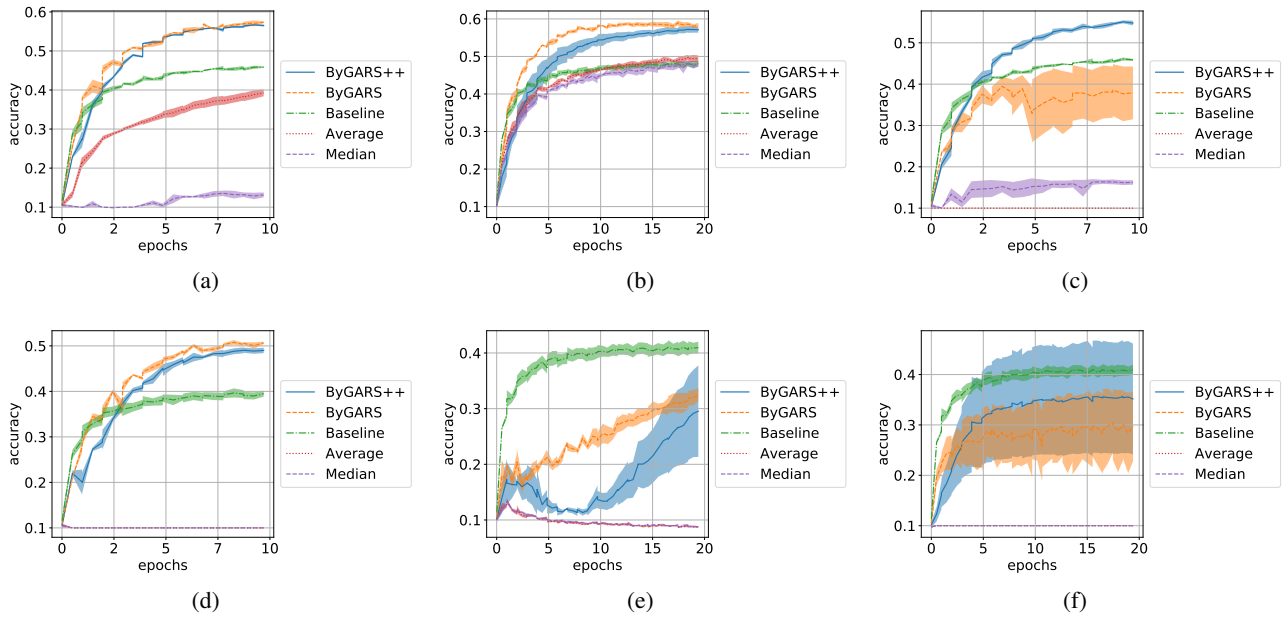

Figure 3: Top-1 accuracy for models trained on CIFAR-10 data under (3a) 3 Inner Product adversaries, (3d) 6 Inner Product adversaries, (3b) 3 Label Flip adversaries, (3e) 6 Label Flip adversaries, (3c) 3 Constant Attack adversaries, and (3f) 6 Constant Attack adversaries. Note the reduction in performance from the top row to the bottom row, due to the increase in the number of adversaries. Despite that, ByGARS, ByGARS++ perform reasonably well.

analysis.

4.4. Distributed Setup

Throughout this section, we will assume a setup with 1 server and 8 workers. We first compare the performance of the algorithms in the absence of any attacks (termed as *No Attack*). In order to show the robustness of our proposed algorithms to an arbitrary number of attackers, we consider multiple cases with different number of attackers, i.e. 3, 6, 8. We consider the case of all 8 adversaries for Sign Flipping attack. As we will see, by allowing negative reputation scores for these workers, our algorithms will achieve similar performance as that of *No Attack*.

4.5. Results and Ablations

We present the results for CIFAR-10 dataset here (refer the website for MNIST and synthetic dataset). Note that we present the mean and standard deviation (shared) of the results for 4 different trials.

An advantage of our proposed methods is that, for a given dataset and model, we used the same learning rate, learning rate decay for all types of attacks, with the only difference being the meta learning rate and meta learning rate decay schedules for ByGARS, and ByGARS++ (see website for details). From this, it is clear that the proposed algorithms are robust to all types of attacks discussed and do not require attack specific information to fine tune the hyper-parameters. Also, from Fig 2 it is evident that there is no trade-off in employing our algorithm in the case of *No Attack*. This shows that our proposed algorithms can serve a much general purpose in distributed learning applications.

We can observe from Figures (3, and figures in main paper here of other CIFAR-10, MNIST and synthetic dataset results) that both ByGARS and ByGARS++ achieve Byzantine robustness against all of the threat models used under varying number of adversaries (with the exception of LIE attack in a few cases). As expected, median fails to defend when the fraction of adversaries is > 0.5 under all attacks. ByGARS, ByGARS++ on the other hand, do not see a lot of degradation wrt *Baseline* when the fraction of adversaries is > 0.5.

We also present results when different types of attacks are carried out at the same time, called as *mixed attack*. From Fig 1, we observe that the proposed algorithms can defend this *mixed attack*. In particular, of the eight workers, only one worker is benign, and the other seven workers have different attack types (local or data poisoning). To the best of our knowledge, we are the first to show robustness against different types of attacks acting at the same time without requiring separate tuning of the parameters to defend against the attacks. The fact that we used the same hyper parameters against all attacks (including the *mixed attack*) makes our

algorithm more applicable to the practical scenarios.

Note that, LIE attack (Baruch et al., 2019) was only defined for fraction of adversaries ≤ 0.5, so we evaluated our algorithms against LIE attack with 3 attackers (< 0.5), and 4 attackers ($= 0.5$). We observe that, ByGARS is robust to 3 attackers for both CIFAR-10 and MNIST datasets, and suffers some degradation in performance when there are 4 attackers. ByGARS++, on the other hand, performs well against 3 attackers on MNIST data, but suffers some degradation on CIFAR-10. In the case of 4 attackers, ByGARS++ fails to defend against the attack. See website for experiments on a synthetic dataset (strongly convex objective) and we can see that both ByGARS, ByGARS++ successfully defend 4 *LIE* (0.5 fraction) adversaries. Also, it is interesting to observe that *Median, Average* (no defense) perform reasonably well against LIE with 3 attackers. This was also observed in (Baruch et al., 2019) and the authors point that LIE is crafted to break defenses like Krum, Bulyan, etc. The authors explain that using plain averaging, the small noise added to the gradients gets averaged out and the impact on the aggregated gradient is minimal. Perhaps, ByGARS performs well against LIE due to the reputation score based aggregation (weighted averaging) as it was observed that the reputation scores were almost equal for all the workers in this case.

It is important to note that, one would expect the *Baseline* to be the best in all scenarios. However, we point out that by using the reputation scores, we are directly affecting the step size of each update performed, and hence it is not surprising to observe that our algorithms perform better than the *Baseline* under *No Attack* or weaker adversary models such as Sign Flip.

Additionally, in order to understand the impact of the auxiliary dataset and the meta updates on the proposed algorithms, we perform an ablation analysis. In particular, we empirically study the approximate size of the auxiliary dataset that is sufficient to achieve convergence of both the proposed algorithms, and we study the number of meta iterations for ByGARS. We chose the label flip attack (with 3 attackers) and fixed it for all the ablation analysis.

4.6. Discussion

In this paper, we proposed two algorithms that compute the reputation scores of workers in a distributed machine learning set up, in order to defend against any number of byzantine adversaries. However, the proposed idea of using reputation scores is much more general as it provides a way to quantify the importance of using a particular local dataset for optimizing a global model even in the non-adversarial case. The reputation scores consistently provide a quantifiable estimate of the utility of a particular worker (and the local dataset at the worker) towards a global optimization problem. This idea can be applicable to several scenarios such as (i) the local datasets are heterogenous and the global optimization objective is to find a model best suited for a particular (unknown) mixture of the local distributions, (ii) the local datasets have varying levels of quality, (iii) the workers have varying levels of privacy constraints, etc. Also, as we observe that the reputation scores for the two proposed algorithms follow different trajectories and have different properties. While ByGARS++ computes reputation scores that start from 0, increase in magnitude and converges to 0 eventually; ByGARS computes reputation scores that grow in magnitude and converges to a fixed (often non-zero) value depending on the attack distribution. We believe that understanding the mechanics of these reputation scores for different distributions of data in the Federated Learning setting is an interesting future direction.

Note that the robustness of our algorithm comes from the fact that we did not design the defense based on any particular criteria such as norm difference, or majority based ideas. We devised the algorithm in order for the optimization to find a descent direction, and hence the superior performance across a range of attacks. However, it is important to note that in this paper, we assumed that the behavior of the attackers is stationary and this limits the ability of the attackers to adapt to the defense algorithm used at the server. With the knowledge of the defense algorithm of the server, a more intelligent adversary can employ a non-stationary attack distribution (simply turn benign when the reputation score is sufficiently negative; and flip back when the reputation score is positive), which is a more challenging problem for the server. We believe that the proposed reputation score based aggregation provides a good platform to address this more setting and we consider this as a potential future direction.

5. Conclusion

We devise a novel class of algorithms based on reputation score based stochastic gradient aggregation for distributed machine learning that is resilient to any number of adversarial workers. The proposed algorithms (ByGARS and ByGARS++) exploit a small auxiliary dataset to compute a reputation score for every worker, and the scores are used to aggregate the workers' gradients. We showed that under reasonable assumptions, ByGARS++ converges to the optimal solution using results from two-timescale stochastic approximation theory (Tadic, 2004). Through simulations, we showed that the proposed algorithms exhibit remarkable robustness property even for non-convex problems under a wide range of Byzantine attacks. Although ByGARS and ByGARS++ are developed for the byzantine attack setting, we believe that these algorithms serve a much general purpose. This algorithm can be modified to train models in other cases such as learning from heterogeneous datasets,

learning under privacy constraints and other adversarial settings (such as adaptive adversaries). We leave such analyses for a future work.

Acknowledgements

The authors thank ARPA-E NEXTCAR program and NSF Grant 1565487 for supporting the research. Results presented in this poster were obtained using the Chameleon testbed supported by the National Science Foundation.

References

Alistarh, D., Allen-Zhu, Z., and Li, J. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 4613–4623, 2018.

Baruch, G., Baruch, M., and Goldberg, Y. A little is enough: Circumventing defenses for distributed learning. In *Advances in Neural Information Processing Systems*, pp. 8635–8645, 2019.

Bernstein, J., Zhao, J., Azizzadenesheli, K., and Anandkumar, A. signSGD with majority vote is communication efficient and fault tolerant. *arXiv preprint arXiv:1810.05291*, 2018.

Blanchard, P., Guerraoui, R., Stainer, J., et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pp. 119–129, 2017.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H. B., et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

Cao, X. and Lai, L. Distributed gradient descent algorithm robust to an arbitrary number of byzantine attackers. *IEEE Transactions on Signal Processing*, 67(22):5850–5864, 2019.

Chang, H., Shejwalkar, V., Shokri, R., and Houmansadr, A. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279*, 2019.

Chee, J. and Toulis, P. Convergence diagnostics for stochastic gradient descent with constant learning rate. In *International Conference on Artificial Intelligence and Statistics*, pp. 1476–1485, 2018.

Chen, L., Wang, H., Charles, Z., and Papailiopoulos, D. Draco: Byzantine-resilient distributed training via redundant gradients. *arXiv preprint arXiv:1803.09877*, 2018.

Chen, Y., Su, L., and Xu, J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–25, 2017.

Damaskinos, G., Mhamdi, E. M. E., Guerraoui, R., Patra, R., and Taziki, M. Asynchronous byzantine machine learning (the case of SGD). *arXiv preprint arXiv:1802.07928*, 2018.

El-Mhamdi, E.-M., Guerraoui, R., Guirguis, A., and Rouault, S. SGD: Decentralized byzantine resilience. *arXiv preprint arXiv:1905.03853*, 2019.

Gupta, N. and Vaidya, N. H. Byzantine fault-tolerant parallelized stochastic gradient descent for linear regression. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 415–420. IEEE, 2019.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

Jin, R., He, X., and Dai, H. Distributed Byzantine tolerant stochastic gradient descent in the era of big data. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6. IEEE, 2019.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

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

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

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

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

Li, L., Xu, W., Chen, T., Giannakis, G. B., and Ling, Q. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1544–1551, 2019.

Portnoy, A. and Hendler, D. Towards realistic Byzantine-robust federated learning. *arXiv preprint arXiv:2004.04986*, 2020.

Tadic, V. B. Almost sure convergence of two time-scale stochastic approximation algorithms. In *Proceedings of the 2004 American Control Conference*, volume 4, pp. 3802–3807. IEEE, 2004.

Xie, C., Koyejo, O., and Gupta, I. Zeno: Byzantine-suspicious stochastic gradient descent. *arXiv preprint arXiv:1805.10032*, 2018.

Xie, C., Koyejo, S., and Gupta, I. Fall of empires: Breaking byzantine-tolerant SGD by inner product manipulation. *arXiv preprint arXiv:1903.03936*, 2019a.

Xie, C., Koyejo, S., and Gupta, I. Zeno++: Robust fully asynchronous SGD. *arXiv preprint arXiv:1903.07020*, 2019b.

Yang, Z. and Bajwa, W. U. ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning. *IEEE Transactions on Signal and Information Processing over Networks*, 5(4):611–627, 2019.

Yang, Z., Gang, A., and Bajwa, W. U. Adversary-resilient inference and machine learning: From distributed to decentralized. *arXiv preprint arXiv:1908.08649*, 2019.

Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.