ROBUST TRAINING IN HIGH DIMENSIONS VIA BLOCK COORDINATE GEOMETRIC MEDIAN DESCENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Geometric median (GM) is a classical method in statistics for achieving a robust estimation of the uncorrupted data; under gross corruption, it achieves the optimal breakdown point of 0.5. However, its computational complexity makes it infeasible for robustifying stochastic gradient descent (SGD) for high-dimensional optimization problems. In this paper, we show that by applying GM to only a judiciously chosen block of coordinates at a time and using a memory mechanism, one can retain the breakdown point of 0.5 for smooth non-convex problems, with non-asymptotic convergence rates similar to that of SGD with GM.[1]

1 INTRODUCTION

Consider smooth non-convex optimization problems with finite sum structure:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \left[\bar{f}(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \right]. \tag{1}$$

Mini-batch SGD is the de-facto method for optimizing such functions (Robbins & Monro, 1951; Bottou, 2010; Tsitsiklis et al., 1986) which proceeds as follows: at each iteration t, it selects a random batch $\mathcal{D}_t$ of b samples, obtains stochastic gradients $\mathbf{g}_t^{(i)} = \nabla f_i(\mathbf{x}_t),\ \forall i \in \mathcal{D}_t$, and updates the parameters using iterations of the form:

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \tilde{\mathbf{g}}^{(t)}, \quad \tilde{\mathbf{g}}^{(t)} = \frac{1}{|\mathcal{D}_t|}\sum_{i\in\mathcal{D}_t} \mathbf{g}_i^{(t)}. \tag{2}$$

In spite of its strong convergence properties in the standard settings (Moulines & Bach, 2011; Dekel et al., 2012; Li et al., 2014; Goyal et al., 2017; Keskar et al., 2016; Yu et al., 2012), it is well known that even a small fraction of corrupt samples can lead SGD to an arbitrarily poor solution (Bertsimas et al., 2011; Ben-Tal & Nemirovski, 2000). This has motivated a long line of work to study robust optimization in presence of corruption (Blanchard et al., 2017; Alistarh et al., 2018a; Wu et al., 2020; Xie et al., 2019). While the problem has been studied under a variety of contamination models, in this paper, we study the robustness properties of the first-order method (2) under the strong and practical **gross contamination model** (See Definition 1) (Li, 2018; Diakonikolas & Kane, 2019; Diakonikolas et al., 2019b;a) which also *generalizes the popular Huber's contamination model and the byzantine contamination framework* (Huber, 1992; Lamport et al., 1982).

In particular, the goal of this work is to design an *efficient* first-order optimization method to solve (1), which remains *robust* even when $0 \leq \psi < 1/2$ fraction of the gradient estimates are *arbitrarily corrupted* in each batch $\mathcal{D}_t$, **without any prior knowledge about the malicious samples**. Note that, by letting the corrupt estimates to be *arbitrarily skewed*, this corruption model is able to capture a number of important and practical scenarios including **corruption in feature** (e.g., existence of outliers) , **corrupt gradients** (e.g., hardware failure, unreliable communication channels during distributed training) and **backdoor attacks** (Chen et al., 2017a; Liao et al., 2018; Gu et al., 2019; Biggio et al., 2012; Mhamdi et al., 2018; Tran et al., 2018).

[1] The implementation of the proposed method is available at Code Link

Algorithm	Iteration Complexity	Breakdown Point
SGD	$\boldsymbol{\mathcal{O}(bd)}$	0
CMD * (Yang et al., 2019; Yin et al., 2018)	$\boldsymbol{\mathcal{O}(bd)}$	1/2 - $\Omega(\sqrt{d/b})$
GMD (Cohen et al., 2016; Wu et al., 2020)	$\mathcal{O}(d\epsilon^{-2} + bd)$	**1/2**
Data & Diggavi (2020)	$\mathcal{O}(db^2 \min(d, b) + bd)$	1/4
BGMD (This work)	$\mathcal{O}(k\epsilon^{-2} + bd)$	**1/2**

Table 1: Comparison of time complexity and robustness properties of different robust optimization methods. The bold quantities show a method achieves the theoretical limits. $*$ CMD throughout the paper will refer to coordinate wise median descent i.e. simply replacing the aggregation step of SGD by CM$(\cdot)$

Definition 1 (**Gross Corruption Model**). *Given $0 \leq \psi < \frac{1}{2}$ and a distribution family $\mathcal{D}$ on $\mathbb{R}^d$ the adversary operates as follows: n samples are drawn from $D \in \mathcal{D}$. The adversary is allowed to* ***inspect*** *all the samples and replace up to ψn samples with arbitrary points.*

Intuitively, this implies that $(1 - \psi)$ fraction of the training samples are generated from the true distribution (*inliers*) and rest are allowed to be **arbitrarily corrupted** (*outliers*) i.e. $\alpha := |\mathbb{B}|/|\mathbb{G}| < 1$, where $\mathbb{B}$ and $\mathbb{G}$ are the sets of corrupt and good samples. In the rest of the paper, we will refer to a set of samples generated through this process as α**-corrupted**.

Definition 2 (**Breakdown Point**). *Finite-sample breakdown point* (Donoho & Huber, 1983) *is a way to measure the resilience of an estimator. It is defined as the smallest fraction of contamination that must be introduced to cause an estimator to break i.e. produce arbitrarily wrong estimates.*
In the context Definition 1 we can say an estimator has the optimal breakdown point 1/2 if it is robust in presence of α-corruption $\forall\ \psi < 1/2$ or alternatively $\forall\ \alpha < 1$.

Definition 3 (**Geometric Median**). *Given a finite collection of observations $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n$ defined over a separable Hilbert space $\mathbb{X}$ with norm $\|\cdot\|$ the geometric median or the Fermet-Weber point* (Haldane, 1948; Weber et al., 1929; Minsker et al., 2015; Kemperman, 1987) *is defined as:*

$$\mathbf{x}_* = \text{GM}(\{\mathbf{x}_i\}) = \arg\min_{\mathbf{y} \in \mathbb{X}} \left[g(\mathbf{x}) := \sum_{i=1}^{n} \|\mathbf{y} - \mathbf{x}_i\| \right] \tag{3}$$

*We call a point $\mathbf{x} \in \mathbb{R}^d$ an ϵ***-accurate geometric median** *if $g(\mathbf{x}) \leq (1 + \epsilon) g(\mathbf{x}_*)$.*

Robust SGD via Geometric Median Descent (GMD). Under the gross corruption model (Definition 1), the vulnerability of mini-batch SGD can be attributed to the linear gradient aggregation step (2) (Blanchard et al., 2017; Alistarh et al., 2018a; Yin et al., 2018; Xie et al., 2019). In fact, it can be shown that *no linear gradient aggregation strategy* can tolerate even a *single* grossly corrupted update, i.e., they have a *breakdown point* (Definition 2) of 0. To see this, consider the single malicious gradient $\mathbf{g}_j^{(t)} = -\sum_{i \in \mathcal{D}_t \setminus j} \mathbf{g}_i^{(t)}$ which results in the average to become $\mathbf{0}$ implying mini-batch SGD getting stuck at the initialization. Motivated by this, a popular approach for robust optimization is to find an estimate $\tilde{g}^{(t)}$ such that with high probability $\|\tilde{g}^{(t)} - \frac{1}{\mathbb{G}} \sum_{\mathbf{g}_i^{(t)} \in \mathbb{G}} \mathbf{g}_i^{(t)}\|$ is small even in presence of gross-corruption (Diakonikolas & Kane, 2019). In this context, geometric median (GM) (Definition 3) is a well studied rotation and translation invariant robust estimator with **optimal breakdown point** of 1/2 even under gross corruption (Lopuhaa et al., 1991; Minsker et al., 2015; Cohen et al., 2016). Due to this strong robustness property, SGD with GM-based gradient aggregation (GMD) has been widely studied in robust optimization literature (Alistarh et al., 2018a; Chen et al., 2017b; Wu et al., 2020). Following the notation of (2) the update step of GMD can be written as:

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \tilde{\mathbf{g}}^{(t)}, \quad \tilde{\mathbf{g}}^{(t)} = \text{GM}(\{\mathbf{g}_i^{(t)}\})\ \forall i \in [b] \tag{4}$$

Despite the strong robustness guarantees of GM, the computational cost of calculating ϵ approximate GM is prohibitive, especially in high dimensional settings. For example, **the best known result (Cohen et al., 2016) uses a subroutine that needs $O(d/\epsilon^2)$ compute to find an ϵ-approximate GM**. Despite recent efforts (Vardi & Zhang, 2000; Weiszfeld, 1937; Chandrasekaran & Tamir, 1989; Chin et al., 2013; Cohen et al., 2016; Pillutla et al., 2019) to design computationally tractable GM$(\cdot)$; given that in practical large-scale optimization settings such as training deep learning models the number of parameters (d) is large (e.g., $d \approx 60M$ for AlexNet, $d \approx 175B$ for GPT-3); GMD remains prohibitively expensive and with limited applicability.

Overview of Our Algorithm (BGMD). In this work, we leverage coordinate selection strategies to significantly reduce the cost of GMD and establish Block co-ordinate GM Descent (BGMD) (Algorithm 1) resulting in nearly *two orders of magnitude* speedup over GMD on most standard deep learning training tasks, while maintaining the same level of accuracy and optimal breakdown point 1/2 even in presence of gross corruption.

At a high level, at each iteration BGMD selects a block of $0 < k \leq d$ *important* coordinates of the stochastic gradients. Importance of a coordinate is measured according to the largest directional derivative measured by the squared ℓ_2 norm across all the samples (Algorithm 2). The remaining $(d-k)$ dimensions are discarded and gradient aggregation happens only along these selected k directions. This Implies the GM subroutine is performed only over gradient vectors in $\mathbb{R}^k$ (a significantly lower dimensional subspace). Thus, when $k \ll d$, this approach provides a practical solution to deploy GM-based aggregation in high dimensional settings.[2] The intuition behind our approach is that as a consequence of over-parameterization, for deep learning models most of the information in the gradients is captured by a small subset of the coordinates Shi et al. (2019). Hence, by the judicious block coordinate selection subroutine outlined in Algorithm 2 one can identify an informative low-dimensional representation of the gradients and use highly robust estimators such as GM even in high dimensional setting which was previously intractable. While Algorithm 2 identifies a representative block of the coordinates, **aggressively reducing the dimension** (i.e., $k \ll d$) might lead to a significant approximation error, which in turn might lead to slower convergence (Nesterov, 2012; Nutini et al., 2015) rate, dwarfing the benefit from reduction in per iteration cost. To alleviate this issue, by leveraging the idea of Error Compensation (Seide et al., 2014; Stich & Karimireddy, 2019; Karimireddy et al., 2019b) we introduce a *memory augmentation* mechanism. Specifically, at each iteration the residual error from dimensionality reduction is computed and accumulated in a memory vector $\hat{\mathbf{m}}_t$ and in the subsequent iteration, $\hat{\mathbf{m}}_t$ is added back to the new gradient estimates.

Algorithm 1 Block GM Descent (BGMD)

Initialize: estimate: $\mathbf{x}_0 \in \mathbb{R}^d$, step-size: γ, memory: $\hat{\mathbf{m}}_0 = \mathbf{0}$, Block Coordinate Selection operator: $\mathcal{C}_k(\cdot)$, Geometric Median operator: GM$(\cdot)$

for *epochs* $t = 0, \ldots,$ *until convergence* **do**

Select samples $\mathcal{D}_t = \{i_1, \ldots, i_b\}$ and obtain : $\mathbf{g}_t^{(i)} := \nabla f_i(\mathbf{x}_t),\ \forall i \in \mathcal{D}_t$

Let $\mathbf{G}_t \in \mathbb{R}^{b \times d}$ s.t. each row $\mathbf{G}_t[i,:] = \mathbf{g}_t^{(i)}$

$\mathbf{G}_t[i,:] \leftarrow \gamma \mathbf{G}_t[i,:] + \hat{\mathbf{m}}_t\ \forall i \in [b]$ (Memory Augmentation)

$\mathbf{\Delta}_t := \mathcal{C}_k(\mathbf{G}_t) \in \mathbb{R}^{b \times k}$ (Select k important dimensions via Algo. 2)

$\mathbf{M}_{t+1} = \mathbf{G}_t - \mathbf{\Delta}_t$ (Compute Residuals)

$\hat{\mathbf{m}}_{t+1} = \frac{1}{b} \sum_{0 \leq i \leq b} \mathbf{M}_{t+1}[i,:]$ (Update memory)

$\tilde{\mathbf{g}}_t := \text{GM}(\mathbf{\Delta}_t)$ (Robust Aggregation in $\mathbb{R}^k$)

$\mathbf{x}_{t+1} := \mathbf{x}_t - \tilde{\mathbf{g}}_t$ (Global model update)

end

Algorithm 2 Block Coordinate Selection Strategy

Input: $\mathbf{G}_t \in \mathbb{R}^{n \times d},\ k$

for *coordinates* $j = 0, \ldots, d\text{-}1$ **do**

$s_j \leftarrow \|\mathbf{G}_t[:,j]\|^2$ (norm along each dimension)

end

Sample set $\mathbb{I}_k$ of k dimensions with probabilities proportional to s_j

$\mathcal{C}_k(\mathbf{G}_t)[i, j \in \mathbb{I}_k] = \mathbf{G}_t[i,j],\ \mathcal{C}_k(\mathbf{G}_t)[i, j \notin \mathbb{I}_k] = 0$

Return: $\mathcal{C}_k(\mathbf{G}_t)$

Contributions. The main contributions of this work are as follows:

- We propose BGMD (Algorithm 1), a method for robust optimization in high dimensions. BGMD is significantly more efficient than the standard GM-SGD method but is still able to maintain the optimal breakdown point 1/2.

[2] The notation $k \ll d$ implies that k is at least an order of magnitude smaller than d.

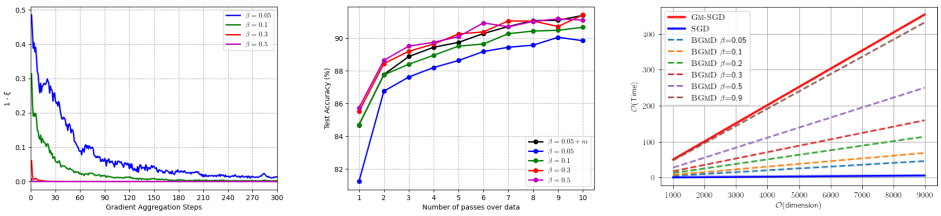
- We provide strong guarantees on the convergence rate of BGMD in standard non-convex scenarios including smooth non-convex functions, and non-convex functions satisfying the Polyak-Łojasiewicz Condition. These rates are comparable to those for GM-SGD under more restricting conditions such as strong convexity Chen et al. (2017b); Pillutla et al. (2019); Wu et al. (2020).
- Through a computational complexity analysis and extensive experiments under several common corruption settings ; we demonstrate that BGMD can be up to 3x more efficient to train than GM-SGD on Fashion MNIST and CIFAR-10 benchmarks while still ensuring similar test accuracy and maintaining same level of robustness.

2 Related Work

Robust optimization in the presence of gross corruption has received renewed impetus in the machine learning community, following practical considerations such as preserving the privacy of the user data and coping with the existence of adversarial disturbances. There are two main research directions in this area: The first direction aims at designing robustness criteria to identify and subsequently filter out corrupt samples before employing the linear gradient aggregation technique in (2). For example, (Ghosh et al., 2019; Gupta et al., 2020) remove the samples with gradient norms exceeding a predetermined threshold; (Yin et al., 2018) remove a fraction of samples from both tails of the gradient norm distribution; (Chen et al., 2018; Yang & Bajwa, 2019) use redundancy (Von Neumann, 1956) and majority vote operations; (Diakonikolas et al., 2019b) rely on spectral filtering; (Steinhardt et al., 2017; Blanchard et al., 2017; Data & Diggavi, 2020; Bulusu et al., 2020) use (ϵ, σ)-resilience based iterative filtering approach; while (Xie et al., 2019) instead add a resilience-based penalty to the optimization task to implicitly remove the corrupt samples. Our approach falls under the second research direction, where the aim is to replace mini-batch averaging with a **robust gradient aggregation operator**. In addition to GM operator (Feng et al., 2014; Alistarh et al., 2018a; Chen et al., 2017b) which was discussed earlier, other examples of robust aggregation techniques including krum (Blanchard et al., 2017), coordinate wise median (Yin et al., 2018) use some approximation of median in high dimensions by loosing some factor in resilience (breakdown point). We also compare a number of robust optimization methods with BGMD in terms of computational complexity and breakdown in Table 1. Please see A.2 for more discussion on connection to prior art.

3 Block Coordinate Geometric Median Descent (BGMD)

As discussed earlier, BGMD (Algorithm 1) involves two key steps: (i) Selecting a block of informative coordinates and run computationally expensive GM aggregation over a low dimensional subspace and (ii) Compensating for the residual error due to block coordinate selection. In the rest of this section we discuss these two ideas in more detail.



(a) Residual Error (b) Importance of Memory (c) Computational Complexity

Figure 1: Training LeNet on fashion mnist with the proposed block descent approach (a) we see that the residual error $r_t \to 0$ as training progresses for suitably chosen k. (b) we plot the generalization performance at different β. We see that training with the memory mechanism (m) enjoys the same accuracy while using a much smaller β. (c) We plot the theoretical asymptotic complexity per iteration (Lemma 2) at different dimensions to highlight the trade off and further emphasize of the importance of the memory mechanism.

Block Selection Strategy. The key intuition why we might be able to select a small number of k coordinates for robust gradient estimation is that in practical over-parameterized models, most of the information of the gradient is likely concentrated along a few coordinates (Chaudhari et al., 2019). So, what would be the *best strategy to select the most informative block of coordinates?*

Ideally, one would like to select the best k dimensions that would result in the largest decrease in training loss. However, this task is NP-hard in general (Das & Kempe, 2011; Nemhauser & Wolsey, 1981; Charikar et al., 2000). Instead, we adopt a simple and fast block coordinate selection rule: Consider $\mathbf{G}_t \in \mathbb{R}^{b\times d}$ where each row corresponds to the transpose of the stochastic gradient estimate: $\mathbf{G}_t[i,:] = (\mathbf{g}_i^{(t)})^T \in \mathbb{R}^{1\times d},\ \forall i \in [b]$ where b is the batch size (2). Then, selecting k dimensions is equivalent to selecting k columns of $\mathbf{G}_t$; which we select according to the norm of the columns. That is, we assign a score to each dimension proportional to the ℓ_2 norm (total mass along that coordinate) i.e. $s_j = \|\mathbf{G}_t[:,j]\|^2$, for all $j \in [d]$. We then sample only k coordinates with with probabilities proportional to s_j and discard the rest to find a set ω_k of size k (see Algorithm 2). We show that this method produces a contraction approximation to $\mathbf{G}_t$.

Lemma 1. *Algorithm 2 yields a* **contraction approximation:** $\mathbb{E}\left[\|\mathcal{C}_k(\mathbf{G}_t) - \mathbf{G}_t\|^2 | \mathbf{G}_t\right] \leq (1-\xi)\|\mathbf{G}_t\|^2,\ \frac{k}{d} \leq \xi \leq 1$, *where* $\mathcal{C}_k(\mathbf{G}_t)[i, j \in \omega_k] = \mathbf{G}_t[i,j],\ \mathcal{C}_k(\mathbf{G}_t)[i, j \notin \omega_k] = \mathbf{0}$.

It is also worth noting that without additional distributional assumption on $\mathbf{G}_t$ the lower bound on ξ cannot be improved [3]. However, in practice the gradient vectors are extremely unlikely to be uniform (Alistarh et al., 2018b) and thus BGMD is expected to satisfy Lemma 1 with $\xi \approx 1$ for sufficiently large k. We provide empirical support in Figure 1(a) where we plot relative residual error $r_t = \|\mathbf{G}_t - \mathcal{C}_{\beta d}(\mathbf{G}_t)\|^2 / \|\mathbf{G}_t\|^2$ of our block selection approach for different $0 < \beta \leq 1$

The Memory Mechanism. While descending along only a small subset of k coordinates at each iteration significantly improves the per iteration computational cost (Lemma 2), a smaller value of k would also imply larger gradient information loss i.e., a smaller ξ (Lemma 1). Intuitively, a restriction to a k-dimensional subspace results in a d/k factor increase in the gradient variance (Stich et al., 2018). We mitigate this by adopting the following a memory mechanism: Throughout training, we keep track of the residual errors in $\mathbf{G}_t - \mathcal{C}_k(\mathbf{G}_t)$ via $\hat{\mathbf{m}}_t \in \mathbb{R}^d$ that we call memory. At each iteration t, it simply accumulates the residual error incurred due to ignoring $d-k$ dimensions, averaged over all the samples participating in that round. In the next iteration, $\hat{\mathbf{m}}_t$ is added back to all the the new gradient estimates as feedback. Following our Jacobian notation, the memory update is given as:

$$\textbf{Memory Augmentation} \quad \mathbf{G}_t[i,:] = \gamma \mathbf{G}_t[i,:] + \hat{\mathbf{m}}_t \ \forall i \in [b]$$

$$\textbf{Memory Update} \quad \mathbf{M}_t = \mathbf{G}_t - \mathcal{C}_k(\mathbf{G}_t)\ ; \hat{\mathbf{m}}_{t+1} = 1/b \sum\nolimits_{0\leq i\leq b} \mathbf{M}_t[i,:]$$

Intuitively, as the residual at each iteration is not discarded but rather kept in memory and added back in a future iteration, this ensures similar convergence rates as training in $\mathbb{R}^d$.

3.1 Computational Complexity Analysis

To theoretically understand the overall computational benefit of our proposed scheme we analyze per iteration computational complexity BGMD and other robust aggregation methods as described in Table 1. Consider solving problem (1) using SGD style iterations of the form (2) with $|\mathcal{D}_t| = b$ and $\mathbf{x} \in \mathbb{R}^d$. Note that the difference between the iterations of SGD, GM-SGD, and BGMD is primarily based on how they aggregate the updates communicated by samples participating in training during that iteration. Formally, at iteration t, let τ_t^a denote the time to aggregate the gradients. Also let, τ_t^b denote the time taken to compute the batch gradients (i.e., time to perform back propagation). Thus, the overall complexity of one training iteration is roughly $\mathcal{O}(\tau_t^a + \tau_t^b)$. Now, note that τ_t^b is approximately the same for all the algorithms mentioned above and for methods like GM-SGD $\tau_t^b \ll \tau_t^a$. So we focus on τ_t^a to study the relative computational cost of different robust gradient aggregation schemes as summarized in Table 1. The complexity proofs are provided in supplementary A.4. In particular, we show that τ_t^a for BGMD is $\mathcal{O}(k/\epsilon^2 + bd)$, where the first term in computational complexity is due to computation of GM of gradients in $\mathbb{R}^k$ and the second term is due to the coordinate sampling and memory procedure. Based on this observation, one can derive the following result[4]:

[3] To see this, consider the case where each $\mathbf{g}_t^{(i)}$ is uniformly distributed along each coordinates. Then, the algorithm would satisfy Lemma 1 with $\xi = \frac{k}{d}$. In this scenario, the achievable bound is identical to the bound achieved via choosing the k dimensions uniformly at random

[4] In most practical settings, $b\epsilon^2 \ll 1/F$

Lemma 2. *Let $k \leq \mathcal{O}(1/F - b\epsilon^2) \cdot d$. Then, given an ϵ- approximate* GM *oracle, Algorithm 1 achieves a factor F speedup over* GM-SGD *for aggregating b samples.*

In contrast, GM-SGD and its variants (Alistarh et al., 2018a; Chen et al., 2017b; Byrd et al., 2012) require computing ϵ-approximate GM of b points in $\mathbb{R}^d$ incurring per iteration cost of at least $\mathcal{O}(d/\epsilon^2)$ (Cohen et al., 2016; Pillutla et al., 2019; Alistarh et al., 2018a; Chen et al., 2017b) and can be significantly costlier than usual SGD which needs only $\mathcal{O}(bd)$ computation per iteration. As we also show in Fig. 1(c) that by choosing a sufficiently small block of coordinates i.e. $k \ll d$, BGMD can result in significant savings per iteration over GMD.

3.2 Convergence Guarantees of BGMD

We first briefly recall some related concepts and state our main assumptions.

Assumption 1 (Stochastic Oracle). *Each non-corrupt sample $i \in \mathbb{G}$ is endowed with an unbiased stochastic first-order oracle with bounded variance, i.e.*

$$\mathbb{E}_{z\sim\mathcal{D}_i}[\mathbf{g}_i(\mathbf{x}, z)] = \nabla f_i(\mathbf{x}), \qquad \mathbb{E}_{z\sim\mathcal{D}_i}\|\nabla F_i(\mathbf{x}, z)\|^2 \leq \sigma^2 \tag{5}$$

Assumption 2 (Smoothness). *Each non-corrupt function f_i is L-smooth, $\forall i \in \mathbb{G}$,*

$$f_i(\mathbf{x}) \leq f_i(\mathbf{y}) + \langle \mathbf{x} - \mathbf{y}, \nabla f_i(\mathbf{y})\rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2. \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d. \tag{6}$$

Assumption 3 (Polyak-Łojasiewicz Condition). *The average of non-corrupt functions $f := \frac{1}{\mathbb{G}}\sum_{i\in\mathbb{G}} f_i(\mathbf{x})$ satisfies the Polyak-Łojasiewicz condition (PLC) with parameter μ, i.e.*

$$\|\nabla f(\mathbf{x})\|^2 \geq 2\mu(f(\mathbf{x}) - f(\mathbf{x}^*)),\ \mu > 0 \quad \text{where } \mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}),\ \forall \mathbf{x} \in \mathbb{R}^d. \tag{7}$$

We further assume that the solution set $\mathcal{X}^* \in \mathbb{R}^d$ is non-empty and convex. Also note that $\mu < L$.

We now analyze the convergence properties of BGMD (Algorithm 1) and state the results in Theorem 1 and Theorem 2 for general non-convex functions and functions satisfying PLC, respectively. [5]

Theorem 1 (Smooth Non-convex). *Consider the case where the functions f_i correspond to non-corrupt samples $i \in \mathbb{G}$ are **non-convex** and **smooth** (Assumption 1-2 hold). Run Algorithm 1 with compression factor ξ (Lemma 1), learning rate $\gamma = 1/2L$ and $\epsilon-$approximate* GM$(\cdot)$ *in presence of $\alpha-$corruption (Definition 1) for T iterations. Then for any $\tau \in [T]$ sampled uniformly at random:*

$$\mathbb{E}\|\nabla f(\mathbf{x}_\tau)\|^2 = \mathcal{O}\left(\frac{L(f(\mathbf{x}_0) - f(\mathbf{x}^*))}{T} + \frac{\sigma^2\xi^{-2}}{(1-\alpha)^2} + \frac{L^2\epsilon^2}{|\mathbb{G}|^2(1-\alpha)^2}\right).$$

Theorem 1 and Theorem 2 state that BGMD with a constant step-size convergences to a neighborhood of a first order stationary point. The radius of this neighborhood depends on two terms. The first term depends on the variance of the stochastic gradients as well as the effectiveness of the coordinate selection strategy through ξ. The second term depends on how accurate the GM computation is performed in each iteration.

Theorem 2 (Non-convex under PLC). *Further assume $f = \frac{1}{\mathbb{G}}\sum_{i\in\mathbb{G}} f_i(\mathbf{x})$ satisfies the PLC with parameter μ i.e. Assumption 1-3 hold. Then, after T iterations of* BGMD *with compression factor ξ, learning rate $\gamma = 1/4L$ and $\epsilon-$approximate* GM$(\cdot)$ *oracle in presence of $\alpha-$corruption satisfies:*

$$\mathbb{E}\|\hat{\mathbf{x}}_T - \mathbf{x}^*\|^2 = \mathcal{O}\left(\frac{LR_0}{\mu^2}\left[1 - \frac{\mu}{8L}\right]^T + \frac{\sigma^2\xi^{-2}}{\mu^2(1-\alpha)^2} + \frac{L^2\epsilon^2}{\mu^2|\mathbb{G}|^2(1-\alpha)^2}\right),$$

for a global optimal solution $\mathbf{x}^* \in \mathcal{X}^*$. Here, $\hat{\mathbf{x}}_T := \frac{1}{W_T}\sum_{t=0}^{T-1} w_t\mathbf{x}_t$ with weights $w_t := (1 - \frac{\mu}{8L})^{-(t+1)}$, $W_T := \sum_{t=0}^{T-1} w_t$.

[5] Due to space constraints a clear proof outline (A.8) and the complete proofs (A.13, A.14) are provided in supplementary material.

Furthermore, both terms in the radius depend on α. By noting that the result holds $\forall \alpha := |\mathbb{B}|/|\mathbb{G}| = \frac{\psi}{1-\psi} < 1$ (see Definition 1) we can can establish the following result.

Remark 1 (BGMD Breakdown Point). BGMD *converges to the neighborhood of a stationary point* $\forall 0 \leq \psi < 1/2$ *i.e. has optimal breakdown point of 1/2.*

We note that the convergence rates established match the rate of GM-SGD when the data is **non i.i.d.** (see e.g. (Chen et al., 2017b; Alistarh et al., 2018a; Wu et al., 2020; Data & Diggavi, 2020) and the references therein). Further, compared to the existing analysis that require strong convexity(Alistarh et al. (2018a); Wu et al. (2020); Data & Diggavi (2020)), Theorem 2 only assumes PLC which is a much milder condition. [6]

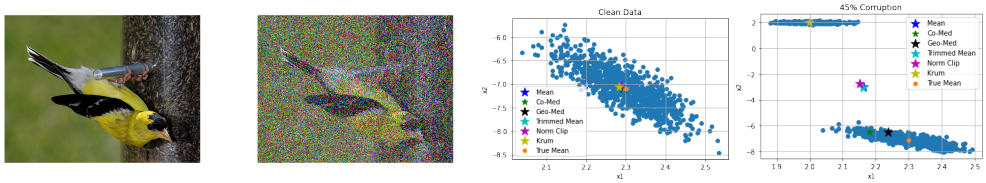
Discussion on choice of β. Based on the discussion above, we note that the size of the block $k = \beta d$ trades off between per-iteration complexity and the final error of the estimate. While a small $0 < \beta \leq 1$ ensures faster iterations (Lemma 2), it also implies that BGMD can converge to a larger neighborhood of sub-optimality (Theorem 1, 2). While finding a bound on β might be difficult since it will depend of the structure of the data itself and thus treated as a hyper-parameter; we will show empirically that it is often possible to run BGMD with a small β to significantly speed up robust optimization while maintaining strong generalization performance (Figure 3, 4 and 5).

4 Empirical Evidence

In this section, we describe our experimental setup, present our empirical findings and establish strong insights about the performance of BGMD. Table 2 provide a summary of the results on two vision datasets comparing BGMD with SGD and other robust aggregation based approaches GMD and CMD. To ensure reproducibility, all the exp. are run with deterministic CUDNN back-end and repeated 5 times with different random seeds and the confidence intervals are noted in the Table 2. In particular, we use the following two important optimization setups for our experiments[7]:

Homogeneous samples. We trained a moderately large LeNet (LeCun et al., 1998) with 1.16M parameters on the challenging Fashion-MNIST Xiao et al. (2017) dataset in the *mini batch* setting with 32 parallel independent and identically distributed (**i.i.d**) mini-batches each with batch size 64. Each experiment under this setting was run for 50 full passes over training data (epochs).

Heterogeneous samples. Our theoretical results (Theorem 1, 2) are **established without any assumption on how the data is distributed across batches**. We verify this by training an 18-layer wide ResNet (He et al., 2016) with 11.2M parameters on CIFAR-10 (Krizhevsky et al., 2012) in a **federated learning** setting (McMahan et al., 2017) with 10 heterogeneous clients each running local SGD with batch size 128. Each experiment was run for 200 epochs.



(a) Original Image (b) Salt & Pepper Noise (c) True gradients (d) 45% corruption

Figure 2: **(a, b) Feature Corruption**: shows the effect of the perturbations added to image. **(c, d) Gradient Corruption**: This Toy example in 2 dimensions visually demonstrates the superior robustness properties of GM for robust mean estimation (e.g. estimating the aggregated gradient) in presence of heavy corruption.

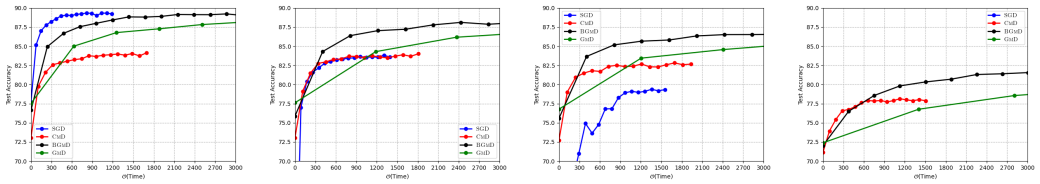
[6] Additionally, in absence of corruption (i.e., $\alpha = 0$), if the data is **i.i.d.**, our theoretical analysis reveals that by setting $\gamma = \mathcal{O}(1/\sqrt{T})$ Algorithm 1 convergences at the rate of $\mathcal{O}(1/\sqrt{T})$ to the statistical accuracy and by setting $\gamma = \mathcal{O}(1/T)$, BGMD convergences at the rate of $\mathcal{O}(\log T/T)$ under PLC. This last result can be established by using the concentration of the median-of-the-means estimator (Chen et al., 2017b).

[7] Please refer to supplementary A.1 for additional results, details on network architecture and hyper-parameter choices and a discussion on comparison with filtering based robust optimization approaches.

In order to simulate a wide variety of real world corruption scenarios we consider all three possible sources of error: corruption in **features**, corruption in **labels**, and corruption in **communicated gradients**. All the experiments are repeated for 0% (i.e. clean), 20% and 40% corruption levels.
Feature Corruption. We consider corruption in the raw training data itself which can arise from different issues related to data collection. Particularly, adopting the corruptions introduced in (Hendrycks & Dietterich, 2018), we use two noise models to directly apply to the corrupt samples:
(i) Additive: Gaussian noise $\mathbf{z}_i \sim \mathcal{N}(0, 100)$ directly added to the image, and
(ii) Impulse: Salt and Pepper noise added by setting 90% of the pixels to 0 or 1.

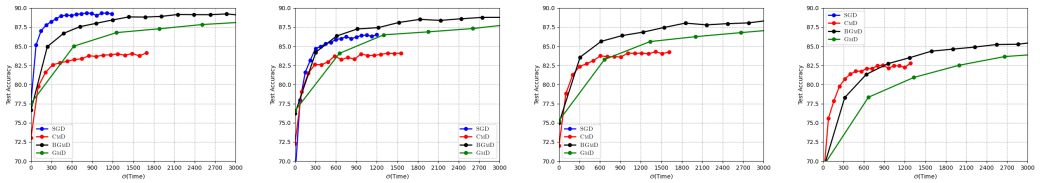
Gradient Corruption. In distributed training over multiple machines the communicated gradients can be noisy, e.g., due to hardware issues or simply because some nodes are adversarial and aim to maliciously disrupt the training. Using standard noise models for gradient corruption (Fu, 1998; Xie et al., 2019; Bernstein et al., 2018) we directly corrupt the gradients in the following manner:
(i) Additive: adversary adds random Gaussian noise $\mathbf{z}_i \sim \mathcal{N}(0, 100)$ to the true gradient, and
(ii)Scaled Bit Flip: corrupted gradients $\mathbf{g}_t^c$ are the scaled bit flipped version of the true gradient (Bernstein et al., 2018) estimate. In particular, we use the following scale: $\mathbf{g}_t^c = -100\mathbf{g}_t$.

Label Corruption. We consider the important backdoor attack (Shen & Sanghavi, 2019; Tolpegin et al., 2020) where the goal of the adversary is to bias the classifier towards some adversary chosen class. To simulate this behavior: at each iteration we flip the labels of randomly chosen ψ fraction of the samples to a **target** label (e.g. in Fashion-MNIST we use 8:bag as the backdoor label).



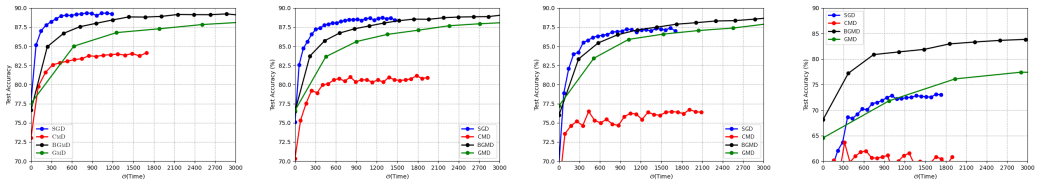
(a) No corruption (b) 10% Corruption (c) 20% Corruption (d) 40% Corruption

Figure 3: **Robustness to Feature Corruption**: Test accuracy of different schemes as a function of **wall clock time** for training Fashion-MNIST using LeNet (i.i.d) in presence of **impulse noise**. Observe that BGMD is able to maintain high accuracy even in presence of strong corruption while attaining at least 3x speedup over GMD whereas CMD performs sub-optimally and SGD diverges at such high level of corruption.



(a) No corruption (b) 10% Corruption (c) 20% Corruption (d) 40% Corruption

Figure 4: **Robustness to Gradient Corruption**: Training Fashion-MNIST using LeNet in i.i.d setting in presence of scaled bit flip corruption to stochastic gradients. Similar to Figure 3, BGMD remains efficient.



(a) No corruption (b) 10% Corruption (c) 20% Corruption (d) 40% Corruption

Figure 5: **Robustness to Label Corruption**: Training Fashion-MNIST (iid) with LeNet in presence of **backdoor attack**. Even in this setting BGMD outperforms SGD, GMD, CMD

Discussion We observe that (Table 2) without corruption both BGMD and GMD are able to achieve similar accuracy as the baseline (i.e., SGD). Conversely, CMD has significant sub-optimality gap even

	Corruption (%)	SGD	CMD	BGMD	GMD
LeNet - Fashion MNIST (homogeneous)					
Clean	-	**89.39**±0.28	83.82±0.26	89.25±0.19	88.98±0.3
Gradient Corruption					
Bit Flip	20	-	84.20±0.02	**88.42**±0.16	88.07±0.05
	40	-	82.33±1.60	**85.67**±0.09	85.57±0.09
Additive	20	-	72.55±0.16	**87.87**±0.33	87.24±0.16
	40	-	41.04±1.13	**88.29**±0.01	83.89±0.08
Feature Corruption					
Additive	20	-	82.38±0.13	**86.76**±0.03	86.63±0.04
	40	-	78.54±0.65	**82.27**±0.06	81.23±0.03
Impulse	20	79.18±6.47	82.59±0.60	**86.91**±0.36	86.23±0.03
	40	-	78.03±0.73	**78.03**±0.73	81.41±0.12
Label Corruption					
Backdoor	20	86.99±0.02	76.38±0.13	**88.97**±0.10	88.26±0.04
	40	73.01±0.68	60.85±1.24	**84.69**±0.31	81.32±0.16
ResNet18 - CIFAR10 (heterogeneous)					
Clean	-	82.29±1.32	85.50±1.43	84.82±0.76	**85.65**±0.48
Gradient Corruption					
Bit Flip	20	-	80.87±0.21	84.56±0.06	**88.07**±0.05
	40	-	77.41±1.04	**82.66**±0.31	80.81±0.01
Additive	20	20.7±1.56	54.75±0.38	**83.84**±0.12	82.40±0.90
	40	-	23.35±6.13	**82.79**±0.68	79.46±0.24

Table 2: Summary of generalization performance under variety of corruption settings. - denotes divergence.

in the clean setting (Chen et al., 2017b). We observe similar trend under different corruption models over various levels of corruption. When corruption is high, SGD starts to diverge after a few iterations. While CMD doesn't diverge, at higher level of corruptions its performance significantly degrades. On the other hand, both GMD and BGMD remain robust and maintain their test accuracy as expected from their strong theoretical guarantees. Surprisingly, **BGMD not only maintains similar robustness as GMD, in several experiments it even outperforms** GMD. In fact, in the heterogeneous setting it outperforms SGD even in absense of corruption. To demonstrate the computational benefit of BGMD we plot test accuracy as a function of the **wall clock time**. Figure 3, 4 and 5 suggest that under a variety of corruption settings BGMD is able to achieve significant speedup over GMD often by more than 3x while maintaining similar (sometime even better) test performance as GMD.

Summary of Results. The above empirical observations highlight the following insights:
(a) For challenging corruption levels/models, GM based methods are indeed superior while standard SGD or CMD can be significantly inaccurate,
(b) In all of our reported experiments, BGMD was run with k set to 10% of the number of parameters. Despite using such a small ratio BGMD retains the strong performance of GMD.
(c) By judiciously choosing k, BGMD is more efficient than GMD, and
(d) memory augmentation is vital for BGMD to attain a high accuracy while employing relatively small values of k.

5 Conclusion

We proposed BGMD, a method for robust optimization in high dimensional setting that achieves the optimal statistical breakdown point while delivering significant savings in the computational costs per iteration compared to existing GM-based strategies. While the current work is more theoretical and algorithmic in nature, we believe that robust learning – the main problem addressed in the work – is a key requirement for deep learning systems to ensure that a few malicious data points do not completely de-rail the system and produce offensive/garbage output.

REFERENCES

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pp. 1709–1720, 2017.

Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 4613–4623, 2018a.

Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, pp. 5973–5983, 2018b.

Zeyuan Allen-Zhu, Zheng Qu, Peter Richtárik, and Yang Yuan. Even faster accelerated coordinate descent using non-uniform sampling. In *International Conference on Machine Learning*, pp. 1110–1119, 2016.

Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, pp. 14668–14679, 2019.

Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization*, 23(4):2037–2060, 2013.

Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical programming*, 88(3):411–424, 2000.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. SignSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569, 2018.

Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3): 334–334, 1997.

Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

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

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Saikiran Bulusu, Prashant Khanduri, Pranay Sharma, and Pramod K Varshney. On distributed stochastic gradient descent for nonconvex functions in the presence of byzantines. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3137–3141. IEEE, 2020.

Richard H Byrd, Gillian M Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1):127–155, 2012.

Ramaswamy Chandrasekaran and Arie Tamir. Open questions concerning weiszfeld's algorithm for the fermat-weber location problem. *Mathematical Programming*, 44(1-3):293–295, 1989.

Moses Charikar, Venkatesan Guruswami, Ravi Kumar, Sridhar Rajagopalan, and Amit Sahai. Combinatorial feature selection problems. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 631–640. IEEE, 2000.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning*, pp. 903–912. PMLR, 2018.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017a.

Yudong Chen, Lili Su, and Jiaming Xu. 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, 2017b.

Hui Han Chin, Aleksander Madry, Gary L Miller, and Richard Peng. Runtime guarantees for regression problems. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pp. 269–282, 2013.

Michael B Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 9–21, 2016.

Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*, 2011.

Deepesh Data and Suhas Diggavi. Byzantine-resilient SGD in high dimensions on heterogeneous data. *arXiv preprint arXiv:2005.07866*, 2020.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *The Journal of Machine Learning Research*, 13:165–202, 2012.

Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Nearest neighbor based greedy coordinate descent. In *Advances in Neural Information Processing Systems*, pp. 2160–2168, 2011.

Ilias Diakonikolas and Daniel M Kane. Recent advances in algorithmic high-dimensional robust statistics. *arXiv preprint arXiv:1911.05911*, 2019.

Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high-dimensions without the computational intractability. *SIAM Journal on Computing*, 48(2):742–864, 2019a.

Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, pp. 1596–1606. PMLR, 2019b.

David L Donoho and Peter J Huber. The notion of breakdown point. *A festschrift for Erich L. Lehmann*, 157184, 1983.

John C Doyle, Bruce A Francis, and Allen R Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.

Jiashi Feng, Huan Xu, and Shie Mannor. Distributed robust learning. *arXiv preprint arXiv:1409.5937*, 2014.

Wenjiang J Fu. Penalized regressions: the bridge versus the lasso. *Journal of computational and graphical statistics*, 7(3):397–416, 1998.

Avishek Ghosh, Raj Kumar Maity, Swanand Kadhe, Arya Mazumdar, and Kannan Ramchandran. Communication-efficient and byzantine-robust distributed learning. *arXiv preprint arXiv:1911.09721*, 2019.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

Nirupam Gupta, Shuo Liu, and Nitin H Vaidya. Byzantine fault-tolerant distributed machine learning using stochastic gradient descent (sgd) and norm-based comparative gradient elimination (cge). *arXiv preprint arXiv:2008.04699*, 2020.

Mert Gurbuzbalaban, Asuman E Ozdaglar, Pablo A Parrilo, and Nuri Denizcan Vanli. When cyclic coordinate descent outperforms randomized coordinate descent. 2017.

JBS Haldane. Note on the median of a multivariate distribution. *Biometrika*, 35(3-4):414–417, 1948.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

Cho-Jui Hsieh and Inderjit S Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1064–1072, 2011.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pp. 492–518. Springer, 1992.

Thorsten Joachims. Making large-scale svm learning practical. Technical report, Technical report, 1998.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *European Conference on Machine Learning and Knowledge Discovery in Databases-Volume 9851*, pp. 795–811, 2016.

Sai Praneeth Karimireddy, Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Efficient greedy coordinate descent for composite problems. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2887–2896. PMLR, 2019a.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pp. 3252–3261. PMLR, 2019b.

JHB Kemperman. The median of a finite measure on a banach space. *Statistical data analysis based on the L1-norm and related methods (Neuchâtel, 1987)*, pp. 217–230, 1987.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

LESLIE Lamport, ROBERT SHOSTAK, and MARSHALL PEASE. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

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

Jerry Zheng Li. *Principled approaches to robust machine learning and beyond*. PhD thesis, Massachusetts Institute of Technology, 2018.

Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. 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.

Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670, 2014.

Cong Liao, Haoti Zhong, Anna Squicciarini, Sencun Zhu, and David Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. *arXiv preprint arXiv:1808.10307*, 2018.

Hendrik P Lopuhaa, Peter J Rousseeuw, et al. Breakdown points of affine equivariant estimators of multivariate location and covariance matrices. *The Annals of Statistics*, 19(1):229–248, 1991.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. *arXiv preprint arXiv:1802.07927*, 2018.

Stanislav Minsker et al. Geometric median and robust estimation in banach spaces. *Bernoulli*, 21(4): 2308–2335, 2015.

Eric Moulines and Francis R Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pp. 451–459, 2011.

Deanna Needell and Joel A Tropp. Paved with good intentions: analysis of a randomized block kaczmarz method. *Linear Algebra and its Applications*, 441:199–221, 2014.

George L Nemhauser and Laurence A Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. In *North-Holland Mathematics Studies*, volume 59, pp. 279–301. Elsevier, 1981.

Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Julie Nutini, Mark Schmidt, Issam Laradji, Michael Friedlander, and Hoyt Koepke. Coordinate descent converges faster with the gauss-southwell rule than random selection. In *International Conference on Machine Learning*, pp. 1632–1641, 2015.

Julie Nutini, Issam Laradji, and Mark Schmidt. Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence. *arXiv preprint arXiv:1712.08859*, 2017.

Sequential Minimal Optimization. A fast algorithm for training support vector machines. *CiteSeerX*, 10(1.43):4376, 1998.

Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Ankan Saha and Ambuj Tewari. On the nonasymptotic convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization*, 23(1):576–601, 2013.

Sylvain Sardy, Andrew G Bruce, and Paul Tseng. Block coordinate relaxation methods for nonparametric wavelet denoising. *Journal of computational and graphical statistics*, 9(2):361–379, 2000.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Shai Shalev-Shwartz and Tong Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pp. 378–385, 2013.

Yanyao Shen and Sujay Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In *International Conference on Machine Learning*, pp. 5739–5748. PMLR, 2019.

Shaohuai Shi, Xiaowen Chu, Ka Chun Cheung, and Simon See. Understanding top-k sparsification in distributed deep learning. *arXiv preprint arXiv:1911.08772*, 2019.

Jacob Steinhardt, Moses Charikar, and Gregory Valiant. Resilience: A criterion for learning in the presence of arbitrary outliers. *arXiv preprint arXiv:1703.04940*, 2017.

Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

Sebastian U Stich. Unified optimal analysis of the (stochastic) gradient method. *arXiv preprint arXiv:1907.04232*, 2019.

Sebastian U Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*, 2019.

Sebastian U Stich, Anant Raj, and Martin Jaggi. Approximate steepest coordinate descent. In *International Conference on Machine Learning*, pp. 3251–3259. PMLR, 2017.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, pp. 4447–4458, 2018.

Nikko Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*, pp. 480–501. Springer, 2020.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, pp. 8000–8010, 2018.

Paul Tseng and Sangwoon Yun. Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *Journal of optimization theory and applications*, 140(3):513, 2009a.

Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1):387–423, 2009b.

John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9): 803–812, 1986.

Yehuda Vardi and Cun-Hui Zhang. The multivariate l1-median and associated data depth. *Proceedings of the National Academy of Sciences*, 97(4):1423–1426, 2000.

John Von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata studies*, 34:43–98, 1956.

Yining Wang and Aarti Singh. Provably correct algorithms for matrix column subset selection with selectively sampled data. *The Journal of Machine Learning Research*, 18(1):5699–5740, 2017.

Alfred Weber, Carl Joachim Friedrich, et al. *Alfred Weber's theory of the location of industries*. The University of Chicago Press, 1929.

Endre Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

Zhaoxian Wu, Qing Ling, Tianyi Chen, and Georgios B Giannakis. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Transactions on Signal Processing*, 68:4583–4596, 2020.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, pp. 6893–6901. PMLR, 2019.

Haibo Yang, Xin Zhang, Minghong Fang, and Jia Liu. Byzantine-resilient stochastic gradient descent for distributed learning: A lipschitz-inspired coordinate-wise median approach. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 5832–5837. IEEE, 2019.

Zhixiong Yang and Waheed U Bajwa. Bridge: Byzantine-resilient decentralized gradient descent. *arXiv preprint arXiv:1908.08098*, 2019.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pp. 5650–5659. PMLR, 2018.

Yang You, Xiangru Lian, Ji Liu, Hsiang-Fu Yu, Inderjit S Dhillon, James Demmel, and Cho-Jui Hsieh. Asynchronous parallel greedy coordinate descent. In *NIPS*, pp. 4682–4690. Barcelona, Spain, 2016.

Hsiang-Fu Yu, Cho-Jui Hsieh, Kai-Wei Chang, and Chih-Jen Lin. Large linear classification when data cannot fit in memory. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4): 1–23, 2012.

Hui Zhang. New analysis of linear convergence of gradient-type methods via unifying error bound conditions. *Mathematical Programming*, 180(1):371–416, 2020.