Robust Training in High Dimensions via Block Coordinate Geometric Median Descent

Anish Acharya
UT Austin

Abolfazl Hashemi
Purdue University

Prateek Jain
Google AI

Sujay Sanghavi
UT Austin

Inderjit Dhillon
UT Austin

Ufuk Topcu
UT Austin

Abstract

Geometric median (GM) is a classical method in statistics for achieving robust estimation of the uncorrupted data; under gross corruption, it achieves the optimal breakdown point of 1/2. However, its computational complexity makes it infeasible for robustifying stochastic gradient descent (SGD) in 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 1/2 for smooth non-convex problems, with non-asymptotic convergence rates comparable to the SGD with GM while resulting in significant speedup in training. We further validate the run-time and robustness of our approach empirically on several popular deep learning tasks. Code available at: https://github.com/anishacharya/BGMD.

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 and Monro, 1951; Bottou, 2010) 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:

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

In spite of its strong convergence properties in the standard settings (Moulines and Bach, 2011), 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 and Nemirovski, 2000). This has motivated a long line of work to study robust optimization in presence of corruption (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 and Kane, 2019; Diakonikolas et al., 2019) which also *generalizes the popular Huber's contamination model (Huber, 1992) and the byzantine contamination framework* (Lamport et al., 1982).

Definition 1 (Gross Corruption Model). *Given $0 \le \psi < \frac{1}{2}$ and a batch $\mathcal{D}_t$ of b samples, 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 samples in a batch 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. Throughout, **we will refer to a set of samples generated through this process as α-corrupted.**

In particular, the goal of this work is to design an *efficient* first-order optimization method to solve (1), which remains *robust* even when $0 \le \psi < 1/2$ frac-

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

Algorithm	Aggregation Operator*	Iteration Complexity†	Breakdown Point†‡
SGD	$\textsc{Mean}(\cdot)$	$\boldsymbol{\mathcal{O}(bd)}$	0
(Yang et al., 2019; Yin et al., 2018)	$\textsc{Cm}(\cdot)$	$\mathcal{O}(bd \log b)$	**1/2**
(Wu et al., 2020)	$\textsc{Gm}(\cdot)$	$\mathcal{O}(d\epsilon^{-2} + bd)$	**1/2**
BGmD (This work)	$\textsc{BGm}(\cdot)$	$\mathcal{O}(k\epsilon^{-2} + bd)$	**1/2**
(Data and Diggavi, 2020)	(Steinhardt et al., 2017)	$\mathcal{O}(db^2 \min(d, b) + bd)$	1/4
(Blanchard et al., 2017)	$\textsc{Krum}(\cdot)$	$\mathcal{O}(b^2 d)$	$\lfloor \beta \rfloor$
(Yin et al., 2018)	$\textsc{CTm}_\beta(\cdot)$	$\mathcal{O}(bd(1-2\beta) + bd \log b)$	$\lfloor \beta \rfloor$
(Ghosh et al., 2019; Gupta et al., 2020)	$\textsc{Nc}_\beta(\cdot)$	$\mathcal{O}(bd(2-\beta) + b \log b)$	$\lfloor \beta \rfloor$

Table 1: Comparison of time-complexity and robustness properties of robust optimization methods (also see Fig. 5) *without any distributional assumptions* on the data. The bold quantities show a method achieves the theoretical limits. The first four methods are related to robust aggregation based approaches while the last four are filtering based approaches.
$*$ Gradient Estimators: $\textsc{Cm}(\cdot)$ co-ordinate wise median, $\textsc{Gm}(\cdot)$ Geometric (spatial) median, $\textsc{BGm}(\cdot)$ Block Geometric Median, $\textsc{CTm}_\beta(\cdot)$ Co-ordinate wise Trimmed mean, $\textsc{Nc}_\beta(\cdot)$ Norm Clipping.
† In section B we discuss the breakdown points and iteration complexities of these methods in more details.
‡ denotes **asymptotic breakdown point**.

tion 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 **corruption in labels** (e.g. label flip (backdoor) attacks).

Robust Gradient Estimation. 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; Yin et al., 2018; Xie et al., 2019). One common approach to measure the resilience of an estimator is through breakdown point (Donoho and Huber, 1983) analysis.

Definition 2 (**Breakdown Point**). *Breakdown point of an estimator is the smallest fraction of contamination that must be introduced to cause an estimator to break i.e. produce arbitrarily wrong estimates.*

In the context of Definition 1 we say an estimator has **optimal breakdown point 1/2** *if it is robust in presence of α-corruption $\forall\ \alpha < 1$.* It can be shown that *no linear gradient aggregation strategy* can tolerate even a *single* grossly corrupted update [1] i.e. they have the **lowest possible asymptotic breakdown of 0**.

Motivated by this, a natural 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.* In the univariate setting, several estimators including median, trimmed mean ($\beta = 0.5$) are known to achieve the optimal breakdown point. A common approach to extend these in multivariate setting is to employ the univariate estimators along each dimension. However when $d > 2$ these estimates need not lie in the convex hull of the samples, are not orthogonal equivariant and can become degenerate (Lopuhaa et al., 1991; Rousseeuw and Leroy, 2005). In this context, spatial estimators like geometric median ($\textsc{Gm}$) (Definition 3) is a well studied rotation and translation invariant robust estimator with **optimal breakdown point** of 1/2 under gross corruption (Minsker et al., 2015; Kemperman, 1987).

Definition 3 (**Geometric Median**). *Given a finite collection of observations $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n$ defined over a separable Hilbert space $\mathbb{X}$ with norm $\|\cdot\|$ the geometric median*(Weber et al., 1929) *is defined as:*

$$\mathbf{x}_* = \textsc{Gm}(\{\mathbf{x}_i\}) = \underset{\mathbf{y} \in \mathbb{X}}{\arg\min} \left[g(\mathbf{x}) := \sum_{i=1}^{n} \|\mathbf{y} - \mathbf{x}_i\| \right] \quad (3)$$

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

Robust SGD via GM Descent (GmD). Due to this strong robustness property, SGD with $\textsc{Gm}$-based gradient aggregation ($\textsc{GmD}$) has been widely studied in robust optimization literature (Alistarh et al., 2018a; Chen et al., 2017; Wu et al., 2020). Following the notation of (2) the update step of $\textsc{GmD}$ is:

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \tilde{\mathbf{g}}^{(t)};\ \tilde{\mathbf{g}}^{(t)} = \textsc{Gm}(\{\mathbf{g}_i^{(t)}\}_{i=1}^{b}) \quad (4)$$

Despite the strong robustness guarantees of $\textsc{Gm}$, the computational cost of calculating ϵ approximate $\textsc{Gm}$ prohibitively expensive and with limited applicability especially in practical high dimensional deep learning settings (Vardi and Zhang, 2000; Weiszfeld, 1937;

[1] 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

Chandrasekaran and Tamir, 1989; Pillutla et al., 2019). 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.

Algorithm 1 Block Gm Descent (BGmD)

```
Initialize: estimate: $\mathbf{x}_0 \in \mathbb{R}^d$, step-size: $\gamma$, memory: $\hat{\mathbf{m}}_0 = \mathbf{0}$, Block Coordinate Selection operator: $\mathcal{C}_k(\cdot)$, Geometric Median operator: $\text{Gm}(\cdot)$
for epochs  t = 0, ..., until convergence do
    select samples $\mathcal{D}_t = \{i_1, \ldots, i_b\}$
    obtain: $\mathbf{g}_t^{(i)} := \nabla f_i(\mathbf{x}_t) \in \mathbb{R}^{1\times d}$, $\forall i \in \mathcal{D}_t$   (back-prop)
    Let $\mathbf{G}_t \in \mathbb{R}^{b\times d}$ s.t. each row $\mathbf{G}_t[i,:] = \mathbf{g}_t^{(i)}$
    $\mathbf{G}_t[i,:] \leftarrow \gamma\mathbf{G}_t[i,:] + \hat{\mathbf{m}}_t \; \forall i \in [b]$   (add memory)
    $\mathbf{\Delta}_t := \mathcal{C}_k(\mathbf{G}_t) \in \mathbb{R}^{b\times k}$   (subset k dim via Algo. 2)
    $\mathbf{M}_{t+1} = \mathbf{G}_t - \mathbf{\Delta}_t$   (compute residuals)
    $\hat{\mathbf{m}}_{t+1} = \frac{1}{b}\sum_{0\le i\le b} \mathbf{M}_{t+1}[i,:]$   (update memory)
    $\tilde{\mathbf{g}}_t := \text{Gm}(\mathbf{\Delta}_t)$  (robust aggregation in $\mathbb{R}^k$)
    $\mathbf{x}_{t+1} := \mathbf{x}_t - \tilde{\mathbf{g}}_t$   (parameter update)
end
```

Algorithm 2 Block Coordinate Selection Strategy

```
Input:  $\mathbf{G}_t \in \mathbb{R}^{n\times d}$, $k$
for coordinates  j = 0, ..., d-1 do
    $s_j \leftarrow \|\mathbf{G}_t[:,j]\|^2$   (norm along each dimension)
end
Choose set $\mathbb{I}_k$ of $k$ dimensions with largest scores $s_j$
$\mathcal{C}_k(\mathbf{G}_t)[i, j \in \mathbb{I}_k] = \mathbf{G}_t[i,j]$, $\mathcal{C}_k(\mathbf{G}_t)[i, j \notin \mathbb{I}_k] = 0$
Return: $\mathcal{C}_k(\mathbf{G}_t)$
```

Overview of Our Algorithm (BGmD). In this work, we leverage coordinate selection strategies to significantly reduce the cost of GmD and establish BGmD (Algorithm 1) resulting in nearly *three 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 under gross corruption.

At a high level, at each iteration BGmD selects a block of $0 < k \le d$ *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 $\mathbb{R}^k$. Thus, when $k \ll d$, this approach provides a practical solution to deploy Gm-based aggregation in high dimensional settings[2]. The intuition 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 and Karimireddy, 2019; Karimireddy et al., 2019b) we introduce the following *memory mechanism*: at each iteration the residual error from dimensionality reduction is computed and accumulated in a memory vector $\hat{\mathbf{m}}_t$ and is added back in the subsequent iteration.

Contributions

- 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.
- 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., 2017; Wu et al., 2020).
- Through 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. Further, in clean setting, we observe that BGmD reaches similar accuracy as SGD while constrained to compute budget (see Fig. 2, 3, 4) indicating BGmD is a practical robust optimization approch in large scale settings.

2 RELATED WORK

Robust optimization in the presence of gross corruption has received renewed impetus in the machine

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

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 (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 and Bajwa, 2019) use redundancy (Von Neumann, 1956) and majority vote operations; (Diakonikolas et al., 2019) rely on spectral filtering; (Steinhardt et al., 2017; Blanchard et al., 2017; Data and Diggavi, 2020; Bulusu et al., 2020) use (ϵ, σ)-resilience based iterative filtering approach; 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., 2017) 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 Supplementary A for more discussion.

3 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.

Block Selection Strategy. The key intuition to *why we might be able to select a small number of coordinates* $k = \beta d,\ \ 0 < \beta \leq 1$ 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 and Kempe, 2011; Nemhauser and 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 stochastic gradient estimate: $\mathbf{G}_t[i,:] = \mathbf{g}_i^{(t)} \in \mathbb{R}^{1\times d},\ \forall i \in [b]$ 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. $s_j = \{\|\mathbf{G}_t[:,j]\|^2\}_{j=1}^d$,.We then sample only k coordinates with with probabilities proportional to s_j and discard the rest to find a set $\mathbb{I}_k$ of size k (see Algorithm 2). The resulting operator $\mathcal{C}_k(\cdot)$ is then: $\mathcal{C}_k(\mathbf{G}_t)[i, j \in \mathbb{I}_k] = \mathbf{G}_t[i,j]$, $\mathcal{C}_k(\mathbf{G}_t)[i, j \notin \mathbb{I}_k] = 0$.

We show $\mathcal{C}_k(\cdot)$ is a contractive mapping to $\mathbf{G}_t$.

Lemma 1. *Algorithm 2 yields a contraction mapping* $\mathbb{E}\left[\|\mathcal{C}_k(\mathbf{G}_t) - \mathbf{G}_t\|^2 | \mathbf{G}_t\right] \leq (1-\xi)\|\mathbf{G}_t\|^2,\ \frac{k}{d} \leq \xi \leq 1.$

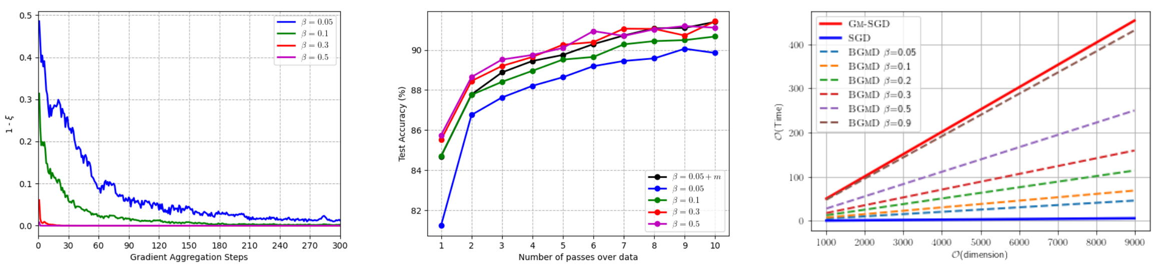
It is worth noting that without additional distributional assumption on $\mathbf{G}_t$ the lower bound on ξ cannot be improved [3]. However, in practice the gradient vectors are very unlikely to be uniform (Alistarh et al., 2018b) and thus BGMD is expected to satisfy Lemma 1 with $\xi \approx 1$ for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

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 memorisation mechanism: Throughout training, we keep track of the residual error $\|\mathbf{G}_t - \mathcal{C}_k(\mathbf{G}_t)\|$ 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:

$$\mathbf{G}_t[i,:] = \gamma \mathbf{G}_t[i,:] + \hat{\mathbf{m}}_t\ \forall i \in [0,b] \tag{5}$$

$$\mathbf{M}_t = \mathbf{G}_t - \mathcal{C}_k(\mathbf{G}_t)\ ; \hat{\mathbf{m}}_{t+1} = \frac{1}{b}\sum_{1\leq i\leq b} \mathbf{M}_t[i,:] \tag{6}$$

[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



(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.

where (5) denotes the **memory augmentation** step and (6) reflects the **memory update** step. Intuitively, as the residual at each iteration is not discarded but rather memorised and added back in a future iteration, this error compensation mechanism ensures similar convergence rates as training in $\mathbb{R}^d$ via decreasing the variance by a constant factor.

3.1 Computational Complexity Analysis

To better 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 to perform back propagation implying 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 in Table 1. Also note that for algorithms like GMD $\tau_t^b \ll \tau_t^a$. So we study τ_t^a for relative computational cost of different robust gradient aggregation schemes as summarized in Table 1. The complexity proofs are provided in supplementary B and E .

In particular, we show that τ_t^a for BGMD is $\mathcal{O}(k/\epsilon^2 + bd)$, where the first term 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. In contrast, GM-SGD and its variants (Alistarh et al., 2018a; Chen et al., 2017; 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., 2017) and can be significantly costlier than usual SGD which needs only $\mathcal{O}(bd)$ computation per iteration. Based on this observation, one can establish the following result [4]:

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

3.2 Discussion on choice of block size.

Based on the discussion above, note that the size of the block $k = \beta d,\ 0 < \beta \le 1$ trades off between per-iteration complexity and the final error of the estimate. While a small β ensures faster iterations (Lemma 2), it also implies that BGMD can converge to a larger neighborhood of sub-optimality (Theorem 1, 2). It is hard to establish a bound on β without additional distributional assumption on the data or structural assumption on the problem and thus treated as a hyper-parameter. However, we empirically show that it is possible to run BGMD with small β to significantly speed up robust optimization while maintaining strong generalization performance (Figure 2, 3 and 4).

3.3 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***:*

$$\mathbb{E}_{z\sim\mathcal{D}_i}[\mathbf{g}_i(\mathbf{x}, z)] = \nabla f_i(\mathbf{x}) \tag{7}$$

$$\mathbb{E}_{z\sim\mathcal{D}_i}\|\nabla F_i(\mathbf{x}, z)\|^2 \le \sigma^2 \tag{8}$$

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

Let $f := 1/\mathbb{G}\sum_{i\in\mathbb{G}} f_i(\mathbf{x})$ denote the average of non-corrupt functions. Then, we also assume that the unconstrained problem $\arg\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$ has a non empty solution set $\mathcal{X}^*$. We will denote the optimal function value as $f(\mathbf{x}^*)$ where $\mathbf{x}^* \in \mathcal{X}^*$ and initial parameters by $\mathbf{x}_0$. For notational convenience define $R_0 = f(\mathbf{x}_0) - f(\mathbf{x}^*)$.

Assumption 2 (Smoothness). *Each non-corrupt function f_i is L-smooth i.e. $\forall i \in \mathbb{G}$ and $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$:*

$$f_i(\mathbf{x}) \le f_i(\mathbf{y}) + \langle \mathbf{x}-\mathbf{y}, \nabla f_i(\mathbf{y})\rangle + \frac{L}{2}\|\mathbf{x}-\mathbf{y}\|^2 \quad (9)$$

Note, if f_i are twice differentiable then this implies that the eigenvalues of $\nabla^2 f_i(\mathbf{x})$ are bounded above by L.

Assumption 3 (Polyak-Łojasiewicz Condition). *f satisfies the Polyak-Łojasiewicz condition (PLC) with parameter $\mu > 0$ (Polyak, 1963):*

$$\|\nabla f(\mathbf{x})\|^2 \ge 2\mu(f(\mathbf{x}) - f(\mathbf{x}^*)),\ \mu > 0 \quad (10)$$

Note, PLC implies: every stationary point is a global minima but doesn't imply uniqueness and thus a milder condition than strong convexity (Karimi et al., 2016).

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 (Satisfying only Assumption 1 and 2) and functions satisfying PLC (i.e. Satisfying Assumptions 1- 3) respectively.

Theorem 1 (Smooth Non-convex). *Suppose Assumption 1-2 hold. Run Algorithm 1 with compression factor ξ (Lemma 1), learning rate $\gamma_t = 1/2L$ 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:*

$$\mathbb{E}\|\nabla f(\mathbf{x}_\tau)\|^2 = \mathcal{O}\left(\frac{LR_0}{T} + \frac{\sigma^2\xi^{-2}}{(1-\alpha)^2} + \frac{L^2\epsilon^2}{|\mathbb{G}|^2(1-\alpha)^2}\right)$$

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

Theorem 2 (Non-convex under PLC). *Suppose Assumption 1-3 hold. Then, after T iterations* BGmD *with compression factor ξ, learning rate $\gamma_t = 1/4L$ and $\epsilon-$approximate* Gm$(\cdot)$ *oracle in presence of $\alpha-$corruption satisfies:*

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

where, $\hat{\mathbf{x}}_T := \frac{1}{W_T}\sum_{t=0}^{T-1} w_t\mathbf{x}_t$, $W_T := \sum_{t=0}^{T-1} w_t$ with weights $w_t := (1-\frac{\mu}{8L})^{-(t+1)}$.

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

Remark 2 (Convergence under i.i.d setting.). *Additionally, if the samples are independent and identically distributed, it is easy to show 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., 2017).*

3.4 Proof Outline

Following Stich (2018); Karimireddy et al. (2019b), we start by defining a sequences of averaged quantities. Divergent from these works however, adopted to the robustness setting: we define these quantities over only the uncorrupted samples (11).

$$\mathbf{g}_t = \frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}}\mathbf{g}_t^i, \quad \bar{\mathbf{g}}_t = \mathbb{E}_t[\mathbf{g}_t] = \frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}}\nabla f_i(\mathbf{x}_t),$$
$$\mathbf{m}_t = \frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}}\mathbf{M}_t[i,:], \quad \Delta_t = \frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}}\Delta_t^i, \quad (11)$$
$$\mathbf{p}_t = \frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}}\mathbf{p}_t^i = \gamma_t\mathbf{g}_t + \mathbf{m}_t.$$

Note that BGmD cannot compute the above average sequences over the uncorrupted samples, but it aims to approximate the aggregated stochastic gradients of the reliable clients, i.e. $\mathbf{g}_t$, via the $\textsc{Gm}(\cdot)$ oracle. Noting the definition of Δ_t in (11), the update can be thought of as being a perturbed sequence with the perturbation quantity $\mathbf{z}_t := \tilde{\mathbf{g}}_t - \textsc{Gm}(\{\Delta_t^i\}_{i\in\mathcal{G}})$. Next, using the closeness of the $\textsc{Gm}(\cdot)$ oracle to the true average we will establish the following lemmas to bound the perturbation.

Lemma 3 (**Bounding the Memory**). *Consider the setting of Algorithm 1 in iteration t with compression factor ξ (Lemma 1), learning rate γ and in presence of $\alpha-$corruption (Definition 1). If further f_i have bounded variance σ^2 (Assumption 1) we have*

$$\mathbb{E}\|\hat{\mathbf{m}}_t\|^2 \le 4(1-\xi^2)\gamma^2\sigma^2\xi^{-2}, \quad \forall i \in [n]. \tag{12}$$

Lemma 4 (**Bounding the Perturbation**). *Consider the setting of Algorithm 1 in iteration t with compression factor ξ (Lemma 1), learning rate γ and $\epsilon-$approximate* $\textsc{Gm}(\cdot)$ *oracle in presence of $\alpha-$corruption (Definition 1). Under the assumption that function f_i are smooth (Assumption 2),*

$$\mathbb{E}\|\mathbf{z}_t\|^2 \le \frac{96\gamma^2\sigma^2}{(1-\alpha)^2}\left[1+\frac{4(1-\xi^2)}{\xi^2}\right] + \frac{2\epsilon^2}{|\mathbb{G}|^2(1-\alpha)^2} \tag{13}$$

With the bound in Lemma 4, we define the following perturbed virtual sequences for $i \in \mathbb{G}$

$$\begin{aligned} \tilde{\mathbf{x}}_{t+1}^i &= \tilde{\mathbf{x}}_t - \gamma\mathbf{g}_t^i - \mathbf{z}_t, \quad \tilde{\mathbf{x}}_0^i = \mathbf{x}_0, \\ \tilde{\mathbf{x}}_{t+1} &= \frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}}\tilde{\mathbf{x}}_{t+1}^i = \tilde{\mathbf{x}}_t - \gamma\mathbf{g}_t - \mathbf{z}_t \end{aligned} \tag{14}$$

Again, $\tilde{\mathbf{x}}_t$ can be though of as a perturbed version of the SGD iterates over only the good samples $i \in \mathbb{G}$. Notice that B$\textsc{Gm}$D doesn't compute the virtual sequence and this sequence is defined merely for the theoretical analysis. Therefore, it is essential to establish its relation to $\mathbf{x}_t$, i.e. the iterates of B$\textsc{Gm}$D. We do so in Lemma 5.

Lemma 5 (**Memory as a Delayed Sequence**). *Consider Algorithm 1 in iteration t. It holds that* $\mathbf{x}_t - \tilde{\mathbf{x}}_t = \mathbf{m}_t$.

The main challenge in showing this result is the presence of perturbations $\mathbf{z}_t$ in the resilient aggregation that we adopt in Algorithm 1. Upon establishing this lemma, using smoothness we establish a bound on suboptimality of the model at each iteration of B$\textsc{Gm}$D as a function of the perturbed virtual sequence $\tilde{\mathbf{x}}_t$.

Lemma 6 (**Recursive Bounding of the Suboptimality**). *For any $0 < \rho < 0.5$ it holds that*

$$\begin{aligned} \mathbb{E}_t[f(\tilde{\mathbf{x}}_{t+1})] \le f(\tilde{\mathbf{x}}_t) &- \left(\frac{1}{2}-\rho\right)\frac{\gamma}{2}\|\nabla f(\mathbf{x}_t)\|^2 \\ &+ \frac{3\gamma L^2}{2}\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2 + L\gamma^2\mathbb{E}_t\|\mathbf{g}_t\|^2 \\ &+ \left(L + \frac{1}{2\rho\gamma} + \frac{1}{2\gamma}\right)\mathbb{E}_t\|\mathbf{z}_t\|^2. \end{aligned} \tag{15}$$

The proofs are furnished by noting that the amount of perturbation can be bounded by using smoothness and the PLC assumptions.

4 EMPIRICAL EVIDENCE

In this section, we describe our experimental setup, present our empirical findings and establish strong insights about the performance of B$\textsc{Gm}$D. 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. Also, in all the experiments BGmD was run using $\beta \le 0.15$. Table 2 provides a summary of the results on two vision datasets. Supplementary C provides additional results, detailed hyper-parameter choices and further discussion. We use the following two important optimization setups:

Homogeneous Distributed Training. We trained 1.16M parameter CNN (LeNet (LeCun et al., 1998)) on Fashion-MNIST (Xiao et al., 2017) dataset 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 Distributed Training. 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) over 10 heterogeneous batches with batch size 128. Following the setup in (Li et al., 2018, 2019b; Das et al., 2020; Karimireddy et al., 2020) we distribute the data among clients in a way such that each client has data from only a few classes. The exact client sampling procedure is described in C. Each experiment was run for 200 epochs.

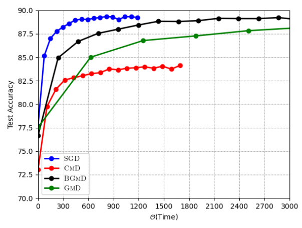
4.1 Corruption Simulation.

We consider three possible sources of error: corruption in **features**, **labels** and **communicated gradients**. All the experiments are repeated for 0% (i.e. clean), 20% and 40% corruption levels.

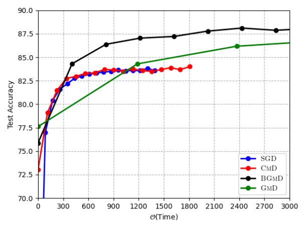
Feature Corruption. We consider corruption in the raw training data itself which can arise from different issues related to data collection. Particularly, adopting the corruptions introduced in (Hendrycks and Dietterich, 2018) we apply the following noise models to the corrupt samples:
(Additive) $\mathbf{z}_i \sim \mathcal{N}(0, 100)$ added to the image.
(Impulse) Salt and Pepper noise added by setting 90% of the pixels to 0 or 1.

	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	**82.11**±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	87.56±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

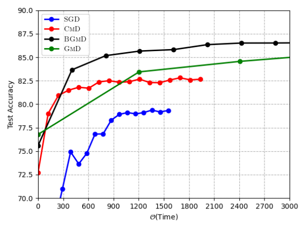
Table 2: Summary of generalization performance under variety of corruption settings. Missing values (-) denotes that the training has diverged. It is clear, that in addition to being efficient BGMD also enjoys superior generalization performance. While, this is an interesting future work, it is possible that the resulting jacobian compression operator $\mathcal{C}_k(\cdot)$ via Algorithm 2 results in implicit regularization benefits in high dimensional settings (Gower et al., 2020; Wu et al., 2019) explaining the superior performance.



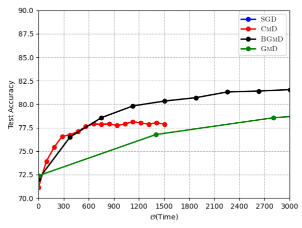
(a) No corruption



(b) 10% Corruption

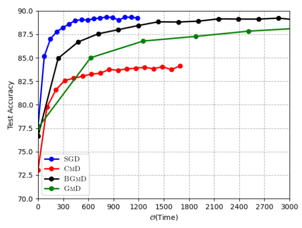


(c) 20% Corruption

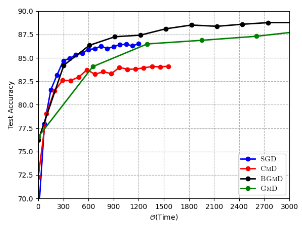


(d) 40% Corruption

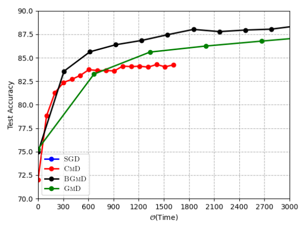
Figure 2: **Robustness to Feature Corruption**: Test accuracy of different schemes as a function of **wall clock time** for training Fashion-MNIST using LeNet (i.i.d) in presence of **impulse noise**. Observe that BGMD is able to maintain high accuracy even in presence of strong corruption while attaining at least 3x speedup over GMD whereas CMD performs sub-optimally and SGD diverges at such levels of corruption. Further, note that in clean setting, BGMD can almost reach the same accuracy of SGD while using the same compute budget. * Note that all the algorithms were run for same number of epochs.



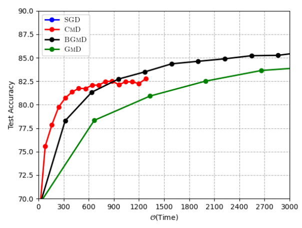
(a) No corruption



(b) 10% Corruption

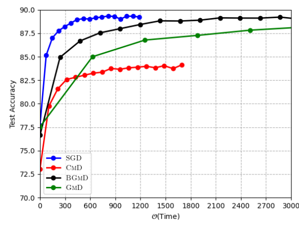


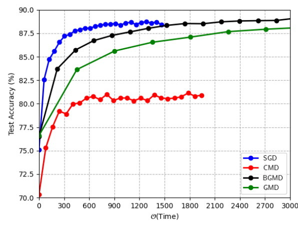
(c) 20% Corruption

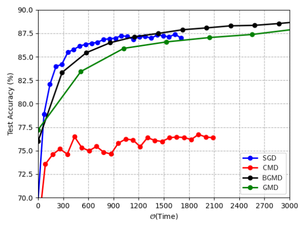


(d) 40% Corruption

Figure 3: **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 2, BGMD remains highly robust. Further, as seen from the plots against **wall clock time** BGMD results in more than 2.5x speedup over all settings. Further in clean setting, BGMD can almost reach the same accuracy of SGD while using the same compute budget.


(a) No corruption


(b) 10% Corruption


(c) 20% Corruption

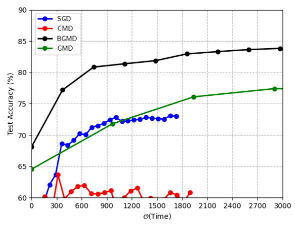
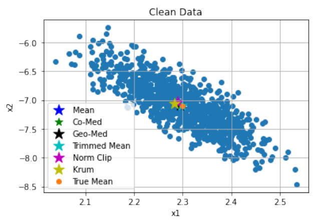

(d) 40% Corruption

Figure 4: **Robustness to Label Corruption**: Training Fashion-MNIST (iid) with LeNet in presence of **backdoor attack**. We note similar superior performance of BGmD while resulting in more than 2.5x speedup over GmD.

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) we directly corrupt the gradients in the following manner: **(Additive)** $\mathbf{z}_i \sim \mathcal{N}(0, 100)$ added to true gradient **(Scaled Bit Flip)** corrupt gradients $\mathbf{g}_t^c$ are the scaled bit flipped version of the true gradient (Bernstein et al., 2018) estimates. In particular: $\mathbf{g}_t^c = -100\mathbf{g}_t$.


(a) True gradients

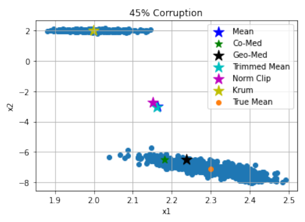

(b) 45% corruption

Figure 5: **Gradient Corruption**: This Toy example in 2 dimensions demonstrates the superior robustness properties of Gm for estimating the aggregated gradient in presence of heavy corruption.

Label Corruption. We consider the important backdoor attack (Shen and Sanghavi, 2019; Liao et al., 2018; Biggio et al., 2012) 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).

4.2 Discussion

We observe that (Table 2) without corruption both BGmD and GmD are able to achieve similar accuracy as the baseline (i.e., SGD) while CmD has significant sub-optimality gap (Chen et al., 2017). 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 but often surpasses the generalization performance over GmD. To demonstrate the computational benefit of BGmD we plot test accuracy as a function of the **wall clock time**. Figure 2, 3 and 4 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. Under clean setting it can reach similar accuracy as SGD while using the same compute budget.

Summary of Results.

A. For challenging corruption levels, Gm based methods are indeed superior while standard SGD or CmD can be significantly inaccurate.

B. By judiciously choosing k, BGmD can often be 3x more efficient than GmD.

C. memory augmentation is crucial for BGmD to attain a high accuracy while using relatively small values of k. Despite using small $\beta \leq 0.15$ in all our experiments, it retains high generalization performance.

5 CONCLUSION

We proposed BGmD, a method for robust, high dimensional optimization that achieves the optimal statistical breakdown point while delivering significant savings in the computational costs per iteration compared to existing Gm-based strategies. BGmD employs greedy coordinate selection and memory augmentation which allows to aggressively select very few coordinates while attaining strong convergence properties comparable to Gm-SGD under standard nonconvex settings. Extensive deep learning experiments demonstrated the efficacy of BGmD.

ACKNOWLEDGEMENTS

This work is supported in part by AFOSR FA9550-19-1-0005, ARL W911NF-17-2-0181, NSF CCF-1564000, NSF IIS-1546452 and NSF HDR-1934932.

References

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

Alistarh, D., Hoefler, T., Johansson, M., Konstantinov, N., Khirirat, S., and Renggli, C. (2018b). The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, pages 5973–5983.

Allen-Zhu, Z., Qu, Z., Richtárik, P., and Yuan, Y. (2016). Even faster accelerated coordinate descent using non-uniform sampling. In *International Conference on Machine Learning*, pages 1110–1119.

Basu, D., Data, D., Karakus, C., and Diggavi, S. (2019). Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, pages 14668–14679.

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

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

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. (2018). SignSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569.

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

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

Biggio, B., Nelson, B., and Laskov, P. (2012). Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*.

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

Blum, M., Floyd, R. W., Pratt, V., Rivest, R. L., and Tarjan, R. E. (1972). Linear time bounds for median computations. In *Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 119–124.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer.

Bruce, D. et al. (2011). A multivariate median in banach spaces and applications to robust pca. *Kättesaadav: http://www-personal. umich. edu/˜ romanv/students/bruce-REU. pdf (26.04. 15)*.

Bulusu, S., Khanduri, P., Sharma, P., and Varshney, P. K. (2020). 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)*, pages 3137–3141. IEEE.

Byrd, R. H., Chin, G. M., Nocedal, J., and Wu, Y. (2012). Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1):127–155.

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

Charikar, M., Guruswami, V., Kumar, R., Rajagopalan, S., and Sahai, A. (2000). Combinatorial feature selection problems. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 631–640. IEEE.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. (2019). Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018.

Chen, L., Wang, H., Charles, Z., and Papailiopoulos, D. (2018). Draco: Byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning*, pages 903–912. PMLR.

Chen, Y., Su, L., and Xu, J. (2017). 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.

Cohen, M. B., Lee, Y. T., Miller, G., Pachocki, J., and Sidford, A. (2016). Geometric median in nearly linear time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 9–21.

Das, A. and Kempe, D. (2011). Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*.

Das, R., Acharya, A., Hashemi, A., Sanghavi, S., Dhillon, I. S., and Topcu, U. (2020). Faster non-convex federated learning via global and local momentum. *arXiv preprint arXiv:2012.04061*.

Data, D. and Diggavi, S. (2020). Byzantine-resilient SGD in high dimensions on heterogeneous data. *arXiv preprint arXiv:2005.07866*.

Dhillon, I. S., Ravikumar, P. K., and Tewari, A. (2011). Nearest neighbor based greedy coordinate descent. In *Advances in Neural Information Processing Systems*, pages 2160–2168.

Diakonikolas, I., Kamath, G., Kane, D., Li, J., Steinhardt, J., and Stewart, A. (2019). Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, pages 1596–1606. PMLR.

Diakonikolas, I. and Kane, D. M. (2019). Recent advances in algorithmic high-dimensional robust statistics. *arXiv preprint arXiv:1911.05911*.

Donoho, D. L. and Huber, P. J. (1983). The notion of breakdown point. *A festschrift for Erich L. Lehmann*, 157184.

Doyle, J. C., Francis, B. A., and Tannenbaum, A. R. (2013). *Feedback control theory*. Courier Corporation.

Feng, J., Xu, H., and Mannor, S. (2014). Distributed robust learning. *arXiv preprint arXiv:1409.5937*.

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

Ghosh, A., Maity, R. K., Kadhe, S., Mazumdar, A., and Ramchandran, K. (2019). Communication-efficient and byzantine-robust distributed learning. *arXiv preprint arXiv:1911.09721*.

Gower, R. M., Richtárik, P., and Bach, F. (2020). Stochastic quasi-gradient methods: Variance reduction via jacobian sketching. *Mathematical Programming*, pages 1–58.

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

Gurbuzbalaban, M., Ozdaglar, A., Parrilo, P. A., and Vanli, N. (2017). When cyclic coordinate descent outperforms randomized coordinate descent. *Advances in Neural Information Processing Systems*, 30.

Haddadpour, F., Kamani, M. M., Mokhtari, A., and Mahdavi, M. (2020). Federated learning with compression: Unified analysis and sharp guarantees. *arXiv preprint arXiv:2007.01154*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hendrycks, D. and Dietterich, T. (2018). Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*.

Hsieh, C.-J. and Dhillon, I. S. (2011). 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*, pages 1064–1072.

Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer.

Joachims, T. (1998). Making large-scale svm learning practical. Technical report, Technical report.

Karimi, H., Nutini, J., and Schmidt, M. (2016). 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*, pages 795–811.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. (2020). Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR.

Karimireddy, S. P., Koloskova, A., Stich, S. U., and Jaggi, M. (2019a). Efficient greedy coordinate descent for composite problems. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2887–2896. PMLR.

Karimireddy, S. P., Rebjock, Q., Stich, S., and Jaggi, M. (2019b). Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR.

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

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

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

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

Li, J. Z. (2018). *Principled approaches to robust machine learning and beyond*. PhD thesis, Massachusetts Institute of Technology.

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

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2018). Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. (2019b). On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*.

Liao, C., Zhong, H., Squicciarini, A., Zhu, S., and Miller, D. (2018). Backdoor embedding in convolutional neural network models via invisible perturbation. *arXiv preprint arXiv:1808.10307*.

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

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

Moulines, E. and Bach, F. R. (2011). Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459.

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

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

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

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

Nutini, J., Schmidt, M., Laradji, I., Friedlander, M., and Koepke, H. (2015). Coordinate descent converges faster with the gauss-southwell rule than random selection. In *International Conference on Machine Learning*, pages 1632–1641.

Olive, D. J. (2001). High breakdown analogs of the trimmed mean. *Statistics & probability letters*, 51(1):87–92.

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

Pillutla, K., Kakade, S. M., and Harchaoui, Z. (2019). Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*.

Polyak, B. T. (1963). Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653.

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

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Rousseeuw, P. J. (1985). Multivariate estimation with high breakdown point. *Mathematical statistics and applications*, 8(37):283–297.

Rousseeuw, P. J. and Leroy, A. M. (2005). *Robust regression and outlier detection*, volume 589. John wiley & sons.

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

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

Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. (2014). 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*.

Shalev-Shwartz, S. and Zhang, T. (2013). Accelerated mini-batch stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 378–385.

Shen, Y. and Sanghavi, S. (2019). Learning with bad training data via iterative trimmed loss minimization. In *International Conference on Machine Learning*, pages 5739–5748. PMLR.

Shi, S., Chu, X., Cheung, K. C., and See, S. (2019). Understanding top-k sparsification in distributed deep learning. *arXiv preprint arXiv:1911.08772*.

Steinhardt, J., Charikar, M., and Valiant, G. (2017). Resilience: A criterion for learning in the presence of arbitrary outliers. *arXiv preprint arXiv:1703.04940*.

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

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

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. (2018). Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, pages 4447–4458.

Stich, S. U. and Karimireddy, S. P. (2019). The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*.

Stich, S. U., Raj, A., and Jaggi, M. (2017). Approximate steepest coordinate descent. In *International Conference on Machine Learning*, pages 3251–3259. PMLR.

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

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

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

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

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

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

Weber, A., Friedrich, C. J., et al. (1929). *Alfred Weber's theory of the location of industries.* The University of Chicago Press.

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

Wu, X., Dobriban, E., Ren, T., Wu, S., Li, Z., Gunasekar, S., Ward, R., and Liu, Q. (2019). Implicit regularization and convergence for weight normalization. *arXiv preprint arXiv:1911.07956*.

Wu, Z., Ling, Q., Chen, T., and Giannakis, G. B. (2020). Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Transactions on Signal Processing*, 68:4583–4596.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xie, C., Koyejo, S., and Gupta, I. (2019). Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, pages 6893–6901. PMLR.

Yang, H., Zhang, X., Fang, M., and Liu, J. (2019). 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)*, pages 5832–5837. IEEE.

Yang, Z. and Bajwa, W. U. (2019). Bridge: Byzantine-resilient decentralized gradient descent. *arXiv preprint arXiv:1908.08098*.

Yin, D., Chen, Y., Kannan, R., and Bartlett, P. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR.

You, Y., Lian, X., Liu, J., Yu, H.-F., Dhillon, I. S., Demmel, J., and Hsieh, C.-J. (2016). Asynchronous parallel greedy coordinate descent. In *NIPS*, pages 4682–4690. Barcelona, Spain.

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

Zuo, Y. (2004). Projection-based affine equivariant multivariate location estimators with the best possible finite sample breakdown point. *Statistica Sinica*, pages 1199–1208.

Supplementary Material for BgmD

A Additional related work

Connection to Coordinate Descent Coordinate Descent (CD) refers to a class of methods wherein at each iteration, a block of coordinates is chosen and subsequently updated using a descent step. While this strategy has a long history (Sardy et al., 2000; Fu, 1998; Bertsekas, 1997; Joachims, 1998; Optimization, 1998), it has received renewed interest in the context of modern large scale machine learning with very high dimensional parameter space (Nesterov, 2012; Richtárik and Takáč, 2014; Tseng and Yun, 2009a,b; Stich et al., 2017; Beck and Tetruashvili, 2013). There has been significant research efforts focusing on efficient coordinate selection strategy. For instance, (Nesterov, 2012; Needell and Tropp, 2014; Shalev-Shwartz and Zhang, 2013; Richtárik and Takáč, 2014; Allen-Zhu et al., 2016) choose the coordinates randomly while (Saha and Tewari, 2013; Gurbuzbalaban et al., 2017) do so cyclically in a fixed order, referred as the Gauss Seidel rule, and (Nutini et al., 2015, 2017; Hsieh and Dhillon, 2011; Dhillon et al., 2011; You et al., 2016; Karimireddy et al., 2019a) propose choosing the coordinates greedily according to norm-based selection criteria, a strategy known as the Gauss Southwell rule.

Remark 3. *Note that, Algorithm 2 is closely related to the Greedy Gauss Southwell coordinate selection approach. In fact, it is immediate that for batch size $b = 1$ Gauss Southwell Co-ordinate Descent becomes a special case of* BGmD.

Connection to Error Feedback. Compensating for the loss incurred due to approximation through a memory mechanism is a common concept in the feedback control and signal processing literature (See (Doyle et al., 2013) and references therein). Seide et al. (2014); Strom (2015) adapt this to gradient compression (1Bit-SGD) to reduce the number of communicated bits in distributed optimization. Recently, (Stich et al., 2018; Stich and Karimireddy, 2019; Karimireddy et al., 2019b) have analyzed this error feedback framework for a number of gradient compressors in the context of communication-constrained distributed training.

Remark 4. *Note that our memory mechanism is inspired by this error feedback mechanism. In fact, all the works on error feedback to compensate for gradient compression* (Stich et al., 2018; Stich and Karimireddy, 2019; Karimireddy et al., 2019b) *are special case of our proposed memory mechanism when batch size $b = 1$ i.e.* $\mathbf{M}_t = \hat{\mathbf{m}}_t$.

B Robust Gradient Estimators

At the heart of BGmD is robust gradient estimation when $0 \leq \psi < 1/2$ fraction of samples are allowed to be grossly corrupted (see Definition 1) i.e. given b gradient vectors $\mathcal{D}_t = \{\mathbf{g}_t^i \in \mathbb{R}^d : \forall i \in [b]\}$, ψb of them are allowed to be arbitrarily corrupted. As discussed in Section 2: one common approach in robust optimization is to replace the mean aggregation by a robust estimator. In this section we elaborate more the robust estimators used in related literature for robust optimization as mentioned earlier in Table 1.

B.1 Univariate Robust Gradient Estimation

Let us first consider the univariate setting (Bruce et al., 2011) i.e. we are given b gradient vectors $\mathcal{D}_t = \{\mathbf{g}_t^i \in \mathbb{R} : \forall i \in [b]\}$. When $\psi = 0$ i.e. in the clean setting: it is very common to use the empirical mean $\mu \in \mathbb{R}$ as the measure of center.

Definition 4 (Mean). *Given $\mathcal{D}_t = \{g_t^i \in \mathbb{R} : \forall i \in [b]\}$, the mean is defined as:*

$$\textsc{Mean}(\{g_t^i \in \mathbb{R}^d : \forall i \in [b]\}) := \mu = \frac{1}{b}\sum_{i\in[b]} g_t^i \tag{16}$$

Another way to define $\textsc{Mean}(\cdot)$ *estimator is via an optimization formulation ; specifically the mean can be defined as the minimizer of the mean squared error as follows:*

$$\mu = \arg\min_{y\in\mathbb{R}} \sum_{i=1}^{b} \|g_t^i - y\|^2 \tag{17}$$

However, as discussed earlier in Section 1, μ is not a robust estimator i.e. when $\psi \neq 0$ the empirical mean μ can be arbitrarily bad. In fact, it can be shown that *no linear gradient aggregation strategy* can tolerate even a *single* grossly corrupted update. It is easy to observe that a single malicious gradient $g_t^j = -\sum_{i\in\mathcal{D}_t\setminus j} g_t^i$ will result in the estimate to become $\mathbf{0}$ implying mini-batch SGD getting stuck at the initialization i.e. they have the lowest possible *finite sample breakdown point* of 1/b or asymptotic breakdown of $\lim_{b\to\infty}\frac{1}{b} = 0$.

In this context, median is a measure which is robust to outliers. In fact, median achieves the ***optimal breakdown point***[5] of **1/2**. The univariate median can be defined as follows:

Definition 5 (Median). *Given $\mathcal{D}_t = \{g_t^i \in \mathbb{R} : \forall i \in [b]\}$, the median is defined as the $\lfloor\frac{b+1}{2}\rfloor$ th order statistic of $\mathcal{D}_t$ is $|\mathcal{D}_t|$ is odd else it is defined as the mean of $\lfloor\frac{b}{2}\rfloor$ th and $\lfloor\frac{b+1}{2}\rfloor$ th order statistics of $\mathcal{D}_t$.*
Another way to define the median estimator $\textsc{Med}(\cdot)$ *is as the minimizer of the sum of absolute errors:*

$$\textsc{Med}(g_t^i : \forall i \in [b]) := \arg\min_{y\in\mathbb{R}} \sum_{i=1}^{b} |g_t^i - y| \tag{18}$$

In addition to having high breakdown point it is often desirable that the robust estimator $T(\cdot)$ is **translation equivariant** i.e. $T(g_t^i + \eta : \forall i \in [b]) = T(g_t^i : \forall i \in [b]) + \eta$ and **permutation equivariant** i.e. $T(g_t^{\pi(i)}) = T(g_t^i : \forall i \in [b])$ for any permutation π (Zuo, 2004; Rousseeuw, 1985). In univariate case, $\textsc{Med}(\cdot)$ satisfies this property.

In the univariate setting, similar to median, another popular L estimator (i.e. relies on ordered statistics) is the trimmed mean

Definition 6 (Trimmed Mean). *Given $\mathcal{D}_t = \{g_t^i \in \mathbb{R} : \forall i \in [b]\}$, $\alpha, \beta \in [0, 1/2)$ the (α, β) trimmed mean $\textsc{Tm}_{(\alpha,\beta)}(\cdot)$ is defined as the average of all the samples in $\mathbb{U}_t \subseteq \mathcal{D}_t$ where $\mathbb{U}_t$ is obtained by removing the β largest and α smallest fraction of samples from $\mathcal{D}_t$. More formally, denoting $\{g_{(1)}, g_{(2)}, \ldots, g_{(b)}\}$ to be the ordered statistics from $\mathcal{D}_t$ i.e. $g_{(1)} \le g_{(2)} \le \cdots \le g_{(b)}$:*

$$\textsc{Tm}_{(\alpha,\beta)}(g_t^i : \forall i \in [b]) := \frac{1}{(1-\alpha-\beta)b} \sum_{j=\lfloor\alpha\rfloor+1}^{\lfloor b(1-\beta)\rfloor} g_{(j)} \tag{19}$$

When $\alpha = \beta$ (Yin et al., 2018) we simply refer to this as β trimmed mean $\textsc{Tm}_\beta(\cdot)$.

Note that, by setting $\beta = b/2$ we can recover the median. It is easy to see that the asymptotic breakdown point of $\textsc{Tm}_{(\alpha,\beta)}(\cdot)$ is $\lfloor\min(\alpha,\beta)\rfloor$ (Olive, 2001).

B.1.1 Multivariate Robust Gradient Estimation

Extending the univariate measures of centers to high dimensions has been a central theme of robust statistics. In particular, it is desired to obtain affine equivariant multivariate estimators with optimal breakdown point 1/2. One approach is to extend the univariate estimators in the multivariate setting by simply applying the univariate

[5]Note that the Breakdown point ϵ^* of an estimator (see Definition 2) can only range between $1/b$ and $1/2$ i.e. $1/b \le \epsilon^* < 1/2$. The lower bound is immediate and the upper bound follows from the fact that the concept of outlier is only applicable when a majority of the points are assumed to be inliers.

estimators co-ordinate wise.

Definition 7 (Co-ordinate wise Median). *Given a set of vectors: $\mathcal{D}_t = \{\mathbf{g}_t^i \in \mathbb{R}^d : \forall i \in [b]\}$ the co-ordinate wise median is defined as the vector $\tilde{\mathbf{g}}_t := \textsc{Cm}(\mathcal{D}_t)$ such that its k-th coordinate is $\tilde{\mathbf{g}}_t[k] := \textsc{Med}(\mathbf{g}_t^i[k] : \forall i \in [b])$ for each co-ordinate $k \in [d]$ where $\textsc{Med}(\cdot)$ is the usual uni-variate median (Definition 5).*

$$\textsc{Cm}(\{\mathbf{g}_t^i \in \mathbb{R}^d : \forall i \in [b]\})[k] = \textsc{Med}(\mathbf{g}_t^i[k] : \forall i \in [b])\ \forall k \in [d] \tag{20}$$

Since, median is an L estimator, it needs to compute the ordered statistics implying $\mathcal{O}(b \log b)$ work per co-ordinate to find the median in one dimension resulting in $\mathcal{O}(bd \log b)$ time in $\mathbb{R}^d$ (Yin et al., 2018) which is quite efficient. In fact, it is also possible to obtain median in linear time (Blum et al., 1972) implying $\textsc{Cm}(\cdot)$ attains the lower bound $\mathcal{O}(bd)$ in $\mathbb{R}^d$. Further, even in multivariate setting the asymptotic breakdown point is maintained at 1/2.

However, in spite of these nice properties, for $d > 2$, $\textsc{Cm}(\cdot)$ does not need to lie in the convex hull of the samples (see (Rousseeuw, 1985) for a great discussion). Empirically, $\textsc{Cm}(\cdot)$ often results in significant regression in generalization performance as we also observed in the experiments, possibly due to the lack of centrality and orthogonal equivariance.

Definition 8 (Co-ordinate wise Trimmed Mean). *Given a set of vectors: $\mathcal{D}_t = \{\mathbf{g}_t^i \in \mathbb{R}^d : \forall i \in [b]\}$ and $\beta \in [0, 1/2)$ β coordinate wise trimmed mean $\textsc{Tm}_\beta(\cdot)$ is simply defined as the vector $\tilde{\mathbf{g}}_t := \textsc{CTm}_\beta(\mathcal{D}_t)$ such that its k-th coordinate is $\tilde{\mathbf{g}}_t[k] := \textsc{Tm}_\beta(\mathbf{g}_t^i[k] : \forall i \in [b])$ for each co-ordinate $k \in [d]$ where $\textsc{Tm}_\beta(\cdot)$ is the usual uni-variate trimmed mean (Definition 6).*

$$\textsc{CTm}_\beta(\{\mathbf{g}_t^i \in \mathbb{R}^d : \forall i \in [b]\})[k] = \textsc{Tm}_\beta(\mathbf{g}_t^i[k] : \forall i \in [b])\ \forall k \in [d] \tag{21}$$

Trimmed mean also requires computing the ordered statistics and then averaging over $(1-2\beta)b$ samples implying computational complexity of $\mathcal{O}((1-2\beta)bd + bd \log b)$. It also maintains **asymptotic breakdown point** $\lfloor \beta \rfloor$. And using the same trick as median (Blum et al., 1972) we can loose the log factor arising from sorting and achieve a $\mathcal{O}(bd)$ algorithm. However, it suffers from the same issues as $\textsc{Cm}(\cdot)$ in terms of centrality and equivariance properties.

Definition 9 (Norm Clipping). *Given a set of vectors: $\mathcal{D}_t = \{\mathbf{g}_t^i \in \mathbb{R}^d : \forall i \in [b]\}$ and $\beta \in [0, 1/2)$ (Ghosh et al., 2019; Gupta et al., 2020) suggest using norm clipping operator $\textsc{Nc}_\beta(\cdot)$ defined as the average of all the samples in $\mathbb{U}_t \subseteq \mathcal{D}_t$ where $\mathbb{U}_t$ is obtained by removing the β fraction of samples with largest norm from $\mathcal{D}_t$. More formally, denoting $\{g_{(1)}, g_{(2)}, \ldots, g_{(b)}\}$ to be the ordered statistics based on norm from $\mathcal{D}_t$ i.e. $\|g_{(1)}\| \le \|g_{(2)}\| \le \cdots \le \|g_{(b)}\|$:*

$$\textsc{Nc}_\beta(\{\mathbf{g}_t^i \in \mathbb{R}^d : \forall i \in [b]\}) = \frac{1}{(1-\beta)b} \sum_{j=1}^{\lfloor b(1-\beta)-1 \rfloor} g_{(j)} \tag{22}$$

We will first note that, in univariate setting this is equivalent to one sided trimmed mean or more formally $(0, \beta)$ trimmed mean (Olive, 2001) and thus **it has asymptotic breakdown point of** $\lfloor \beta \rfloor$. However, (Ghosh et al., 2019) has shown that **it is possible to achieve optimal breakdown point under a very restrictive assumptions on the distribution of the samples - which are hard to be satisfied in real world settings**.

C Additional Experimental Details

In this section, we provide more details on our experimental setup and provide some additional result that we didn't include in the main paper due to space constraint.

C.1 Additional hyper-parameter details

Common to all the setting we used a cross-entropy loss with weight decay 1e-5. For both MNIST and Fashion-MNIST experiments, we use the learning rate schedule suggested in Bottou (2012); Haddadpour et al. (2020)

	Corruption (%)	SGD	CmD	BGmD	GmD
Clean	-	**99.27**±0.01	98.83±0.02	99.09±0.05	99.24±0.02
		Gradient Attack			
Bit Flip	20	9.51±1.77	98.79±0.01	**99.06**±0.02	98.98±0.01
	40	9.60±2.04	93.69±0.09	97.89±0.05	**98.11**±0.12
Additive	20	9.68±0.11	94.26±0.03	98.61±0.01	**98.69**±0.01
	40	9.74±0.12	91.86±0.03	**97.78**±0.27	92.78±0.04

Table 3: Multi Layer Perceptron trained on MNIST in regular i.i.d. setting. For all corruption types, test accuracy of BGmD is similar to that of GmD and surprisingly, in some cases even higher. As expected, SGD fails to make progress under corruption . CmD performs sub-optimally as corruption is increased.

where we reduce the client learning rate by 1% after every round, i.e., $\eta_k = (0.99)^k \eta_0$, η_0 being the initial learning rate. We search the initial learning rates over $\{10^{-3}, 5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$. For the Fashion MNIST experiment we used initial learning rate of 0.01 and for the MNIST experiment we used an initial learning rate of 0.1. For the heterogeneous CIFAR10 experiment we have used ResNet-18 architecture included in torchvision models. We used an initial learning rate of 0.1 with cosine annealing learning rate schedule.

C.2 Heterogeneous Data Distribution Simulation

In order to simulate heterogeneous real world data distribution across nodes for our CIFAR10 experiment: first, the training data was sorted using labels and divided into 40 consecutive equal data-shards. Note that, the sharding procedure ensures each shard only containing data from a single class. Then since we used 10 clients each client was assigned 4 shards sampled uniformly at random without replacement. This implies each client i had access to c_i classes where $1 \leq c_i \leq 4$. Note that in expectation each client should have data from only 4 classes.

Detailed Proofs of BgmD

D Proof of Lemma 1 (Sparse Approximation)

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

Proof. Suppose, $\Omega_k = \{\omega \subseteq \{1,2,...,d\} : |\omega| = k\}$ is the set of all possible subsets of cardinality k i.e. $|\Omega_k| = \binom{d}{k}$. Also let the embeddings produced by random co-ordinate sampling and active norm sampling (Algorithm 2) are denoted by $\mathcal{C}_k^r(\cdot)$ and $\mathcal{C}_k^n(\cdot)$ respectively and let the $i-$th row $\mathbf{G}_t[i,:] = \mathbf{g}_t^i$. Then, we can bound the reconstruction error in expectation $\forall \mathbf{G}_t \in \mathbb{R}^{b\times d}$ as:

$$\begin{aligned}
\mathbb{E}_{\mathcal{C}_k^n}\left[\|\mathcal{C}_k^n(\mathbf{G}_t) - \mathbf{G}_t\|_F^2 | \mathbf{G}\right] &\leq \mathbb{E}_{\mathcal{C}_k^n}\left[\|\mathcal{C}_k^r(\mathbf{G}_t) - \mathbf{G}_t\|^2 | \mathbf{G}_t\right] \\
&= \sum_{i=1}^{n} \mathbb{E}_{\mathcal{C}_k^n}\left[\|\mathcal{C}_k^r(\mathbf{g}_t^i) - \mathbf{g}_t^i\|^2 | \mathbf{g}_t^i\right] \\
&= \sum_{i=1}^{n} \frac{1}{|\Omega_k|} \sum_{\omega\in\Omega_k} \sum_{i=1}^{d} \mathbf{x}_i^2 \mathbb{I}\{i \notin \omega\} \\
&= \sum_{i=1}^{n} (1 - \frac{k}{d}) \|\mathbf{g}_t^i\|^2 \\
&= (1 - \frac{k}{d}) \|\mathbf{G}_t\|_F^2
\end{aligned}$$

■

E Computational Complexity of BGmD iterates

Proposition 1. (Computational Complexity). *Given an ϵ- approximate* Gm *oracle , each gradient aggregation of* BGmD *with block size k incurs a computational cost of: $\mathcal{O}(\frac{k}{\epsilon^2} + bd)$.*

Proof. Let us first recall, one gradient aggregation operation: given b stochastic gradients $\mathbf{g}_i$ $\forall i \in [b]$, the goal of the aggregation step is to compute $\tilde{\mathbf{g}}$ for subsequent iterations of the form $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta\tilde{g}$. Further, $\mathbf{G}_t \in \mathbb{R}^{b\times d}$ denote the gradient jacobian where i th row $\mathbf{G}_t[i,:]$ corresponds to $\mathbf{g}_i$. Also assume that we have at our disposal an oracle Gm$(\cdot)$ that can compute an ϵ-approximate Gm $\mathbf{g}_i \in \mathbb{R}^d$ $\forall i \in [b]$ using $\mathcal{O}(\frac{d}{\epsilon^2})$ compute Cohen et al. (2016). Recall that BGmD (Algorithm 1) is composed of the following main steps:

- **Memory Augmentation**: At each gradient aggregation step, BGmD *needs to add back the stored memory $\hat{\mathbf{m}}_t$ compensating for accumulated residual error incurred in previous iterations to $\mathbf{G}_t$ such that $\mathbf{G}_t[i,:] = \gamma\mathbf{G}_t[i,:] + \hat{\mathbf{m}}_t$. Note that, this is a row wise linear (addition) operation implying $\mathcal{O}(bd)$ associated cost.*
- **Active Norm Sampling**: At each gradient aggregation step: BGmD *selects k of the d coordinates i.e. k columns of $\mathbf{G}_t$ using Algorithm 2. This requires computing the ℓ_2 norm distribution along the d columns, followed by sampling k of them proportional to the norm distribution. The computational complexity of computing Active Norm Sampling is* $\mathcal{O}(bd)$ Wang and Singh (2017).
- **Compute $\mathbf{M}_{t+1}$**: Further, memory needs to be updated to $\mathbf{M}_{t+1}$ for future iterates implying another row wise linear operation incurring $\mathcal{O}(bd)$ compute.
- **Low Rank Gm**: *Note that* BGmD *needs to run* Gm$(\cdot)$ *over $\mathbb{R}^k$ implying a cost of $\mathcal{O}(\frac{k}{\epsilon^2})$.*

Putting it together, the total cost of computing gradient aggregation per iteration using Algorithm 1 is then $\mathcal{O}(\frac{k}{\epsilon^2} + bd)$. ■

E.1 Proof of Lemma 2 (Choice of k)

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

Proof. First, note that: for one step of gradient aggregation Gm-SGD makes one call to Gm$(\cdot)$ oracle implying a $\mathcal{O}(\frac{d}{\epsilon^2})$ computational cost per gradient aggregation.

Now, let us assume, $k = \beta d$ where $0 < \beta \leq 1$ denotes the fraction of total gradient dimensions retained by BGmD. Then using Proposition 1 we can find a bound on β such that gradient aggregation step of BGmD has a linear speedup over that of GM-SGD by a linear factor F.

$$
\begin{aligned}
&\mathcal{O}(\frac{d}{\epsilon^2}) \geq F\cdot\mathcal{O}(\frac{k}{\epsilon^2}+bd)\\
\Rightarrow&\mathcal{O}(\frac{1}{\epsilon^2}) \geq \mathcal{O}(\frac{F\beta}{\epsilon^2})+\mathcal{O}(Fb)\\
\Rightarrow&\mathcal{O}(F\beta) \leq \mathcal{O}(1)-\mathcal{O}(Fb\epsilon^2)\\
\Rightarrow&\mathcal{O}(\beta) \leq \mathcal{O}(\frac{1}{F})-\mathcal{O}(b\epsilon^2)
\end{aligned}
\tag{23}
$$

This concludes the proof. ■

F Detailed Statements of the convergence Theorems

We here state the full version of the main convergence theorems

Theorem 1 (**Smooth Non-convex**). *Consider the general case where the functions f_i correspond to non-corrupt samples $i \in \mathbb{G}$ i.e. $f = \frac{1}{\mathbb{G}}\sum_{i\in\mathbb{G}} f_i(\mathbf{x})$ are* ***non-convex*** *and* ***smooth*** *(Assumption 2). Define, $R_0 := f(\mathbf{x}_0) - f(\mathbf{x}^*)$ where $\mathbf{x}^*$ is the true optima and $\mathbf{x}_0$ is the initial parameters. Run Algorithm 1 with compression factor ξ (Lemma 1), learning rate $\gamma = 1/2L$ and $\epsilon-$approximate GM$(\cdot)$ oracle in presence of $\alpha-$corruption (Definition 1) for T iterations. Then it holds that:*

$$
\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mathbf{x}_t)\|^2 &\leq \frac{8R_0}{\gamma T} + 8L\gamma\sigma^2 + \frac{48L^2\gamma^2\sigma^2(1-\xi^2)}{\xi^2}\\
&\quad + \frac{2304\sigma^2}{(1-\alpha)^2}\left[1+\frac{4(1-\xi^2)}{\xi^2}\right] + \frac{48\epsilon^2}{\gamma^2|\mathbb{G}|^2(1-\alpha)^2}\\
&= \mathcal{O}\left(\frac{LR_0}{T} + \frac{\sigma^2\xi^{-2}}{(1-\alpha)^2} + \frac{L^2\epsilon^2}{|\mathbb{G}|^2(1-\alpha)^2}\right)
\end{aligned}
\tag{24}
$$

Theorem 2 (**Non-convex under PLC**). *Assume in addition to* ***non-convex*** *and* ***smooth*** *(Assumption 2) the functions f_i correspond to non-corrupt samples also satisfy the* ***Polyak-Łojasiewicz Condition*** *(Assumption 3) with parameter μ. After T iterations Algorithm 1 with compression factor ξ (Lemma 1), learning rate $\gamma = 1/4L$ and $\epsilon-$approximate GM$(\cdot)$ oracle in presence of $\alpha-$corruption (Definition 1) satisfies:*

$$
\begin{aligned}
\mathbb{E}\|\hat{\mathbf{x}}_T - \mathbf{x}^*\|^2 &\leq \frac{16(f(\mathbf{x}_0)-f^*)}{\mu^2\gamma}\left[1-\frac{\mu\gamma}{2}\right]^T + \frac{16L\gamma\sigma^2}{\mu^2} + \left[\frac{80\gamma^2\sigma^2L^2(1-\xi^2)}{\mu^2\xi^2}\right]\\
&\quad + \frac{3072\sigma^2}{\mu^2(1-\alpha)^2}\left[1+\frac{4(1-\xi^2)}{\xi^2}\right] + \frac{64\epsilon^2}{\mu^2\gamma^2|\mathbb{G}|^2(1-\alpha)^2}\\
&= \mathcal{O}\left(\frac{LR_0}{\mu^2}\left[1-\frac{\mu}{8L}\right]^T + \frac{\sigma^2\xi^{-2}}{\mu^2(1-\alpha)^2} + \frac{L^2\epsilon^2}{\mu^2|\mathbb{G}|^2(1-\alpha)^2}\right)
\end{aligned}
\tag{25}
$$

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 $w_t := (1-\frac{\mu}{8L})^{-(t+1)}$, $W_T := \sum_{t=0}^{T-1} w_t$.

G Intermediate Facts and Lemmas

Fact 1. $\mathbb{E}\|\sum_{i\in\mathcal{A}}\mathbf{a}_i\|^2 \leq |\mathcal{A}|\sum_{i\in\mathcal{A}}\mathbb{E}\|\mathbf{a}_i\|^2$
This can be seen as a consequence of the Jensen's inequality.

Fact 2 (**Young's Inequality**). *For any $\beta > 0$,*

$$
\mathbb{E}[\langle\mathbf{a},\mathbf{b}\rangle] \leq \frac{\beta}{2}\mathbb{E}\|\mathbf{a}\|^2 + \frac{1}{2\beta}\mathbb{E}\|\mathbf{b}\|^2 \tag{26}
$$

This can be seen as a special case of the weighted AM-GM inequality.

Fact 3 (**Lemma 2 in Stich (2019)**)**.** *Let $\{a_t\}$ and $\{b_t\}$ be to non-negative sequences such that*

$$a_{t+1} \le (1 - r\gamma)a_t - s\gamma b_t + c, \tag{27}$$

where $r, \gamma, s, c > 0$ and $r\gamma < 1$. Let $w_t = (1 - r\gamma)^{-(t+1)}$ and $W_T = \sum_{t=0}^{T-1} w_t$. Then the following holds:

$$\frac{s}{W_T} \sum_{t=0}^{T-1} b_t w_t + r a_T \le \frac{a_0}{\gamma}(1 - r\gamma)^T + \frac{c}{\gamma}. \tag{28}$$

Fact 4. *Let f be a function that satisfies PLC (Assumption 3) with parameter μ. Then, f satisfies the quadratic growth condition* Karimi et al. (2016); Zhang (2020)*:*

$$f(\mathbf{x}) - f^* \ge \frac{\mu}{2}\|\mathbf{x} - \mathbf{x}_p\|^2 \tag{29}$$

where $\mathbf{x}_p$ is the projection of $\mathbf{x}$ onto the solution set $\mathcal{X}^*$.

G.1 Proof of Lemma 3

Proof. The result is due to Lemma 5 in Basu et al. (2019) which is inspired by Lemma 3 in Karimireddy et al. (2019b). The proof relies on using the fact that the proposed norm sampling operator $\mathcal{C}_k$, as shown in Lemma 1 is contractive. Hence, we can use this property to derive a recursive bound on the norm of the memory. Using this result as well as the bounded stochastic gradient assumption results in a geometric sum that can be bounded by the RHS of (12). ■

G.2 Proof of Lemma 4

Proof. Since $\tilde{\mathbf{g}}_t$ is the ϵ-accurate GM of $\{\Delta_t^i\}_i$, $\mathbf{z}_t$ can be thought of as the ϵ-accurate GM of $\{\Delta_t^i - \Delta_t\}_i$. Thus, by the classical robustness property of GM (Theorem. 2.2 in Lopuhaa et al. (1991); see also Minsker et al. (2015); Cohen et al. (2016); Chen et al. (2017); Li et al. (2019a); Wu et al. (2020) for similar adaptations) we have

$$\mathbb{E}\|\mathbf{z}_t\|^2 \le \frac{8|\mathbb{G}|}{(|\mathbb{G}| - |\mathbb{B}|)^2} \sum_{i\in\mathbb{G}} \mathbb{E}\|\Delta_t^i - \Delta_t\|^2 + \frac{2\epsilon^2}{(|\mathbb{G}| - |\mathbb{B}|)^2}. \tag{30}$$

Next, we bound $\mathbb{E}\|\Delta_t^i - \Delta_t\|^2$ using the properties of memory mechanism stated in Lemma 3

$$\begin{aligned} \mathbb{E}\|\Delta_t^i - \Delta_t\|^2 &= 2\mathbb{E}\|\Delta_t^i\|^2 + 2\mathbb{E}\|\Delta_t\|^2 \\ &\le 2\mathbb{E}\|\Delta_t^i\|^2 + \frac{2}{|\mathbb{G}|}\sum_{i\in\mathbb{G}} \mathbb{E}\|\Delta_t^i\|^2 \\ &\le 4 \max_{i\in\mathbb{G}} \mathbb{E}\|\Delta_t^i\|^2, \end{aligned} \tag{31}$$

where we used Fact 1 twice. Next, we establish a bound on the norm of the communicated messages as follows. Add and subtract $\mathbf{p}_t^i$ and use the update rule of the memory to obtain

$$\begin{aligned} \mathbb{E}\|\Delta_t^i\|^2 &= \mathbb{E}\|\Delta_t^i + \mathbf{p}_t^i - \mathbf{p}_t^i\|^2 \\ &= \mathbb{E}\|\mathbf{p}_t^i - \hat{\mathbf{m}}_{t+1}\|^2 \\ &= \mathbb{E}\|\gamma\mathbf{g}_t^i + \hat{\mathbf{m}}_t - \hat{\mathbf{m}}_{t+1}\|^2 \\ &\le 3\mathbb{E}\|\gamma\mathbf{g}_t^i\|^2 + 3\mathbb{E}\|\hat{\mathbf{m}}_t\|^2 + 3\mathbb{E}\|\hat{\mathbf{m}}_{t+1}\|^2, \\ &\le 3\gamma^2\sigma^2 + 3\mathbb{E}\|\hat{\mathbf{m}}_t\|^2 + 3\mathbb{E}\|\hat{\mathbf{m}}_{t+1}\|^2, \end{aligned} \tag{32}$$

by definition of $\mathbf{p}_t^i$ and Fact 1. Notice that using Lemma 3 we can uniformly bound the last two terms on the RHS of (32):

$$3\mathbb{E}\|\hat{\mathbf{m}}_t\|^2 + 3\mathbb{E}\|\hat{\mathbf{m}}_{t+1}\|^2 \le \frac{24(1 - \xi^2)\gamma^2\sigma^2}{\xi^2}. \tag{33}$$

Therefore, we conclude

$$\mathbb{E}\|\Delta_t^i\|^2 \leq 3\gamma^2\sigma^2 + \frac{24(1-\xi^2)\gamma^2\sigma^2}{\xi^2}. \tag{34}$$

Therefore, by (31) and (34)

$$\mathbb{E}\|\Delta_t^i - \Delta_t\|^2 \leq 12\gamma^2\sigma^2 \left[1 + \frac{4(1-\xi^2)}{\xi^2}\right], \tag{35}$$

and the proof is complete by combining this last result with (30). ■

G.3 Proof of Lemma 5

Proof. We derive a recursive relation for the difference $\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}$. It follows that

$$\begin{aligned}\mathbf{x}_{t+1} &= \mathbf{x}_t - \tilde{\mathbf{g}}_t \\ &= \mathbf{x}_t - \Delta_t - \mathbf{z}_t.\end{aligned} \tag{36}$$

On the other hand,

$$\begin{aligned}\tilde{\mathbf{x}}_{t+1} &= \tilde{\mathbf{x}}_t - \gamma\mathbf{g}_t - \gamma\mathbf{z}_t \\ &= \tilde{\mathbf{x}}_t - \gamma\mathbf{g}_t - \mathbf{z}_t \\ &= \tilde{\mathbf{x}}_t - \gamma\mathbf{g}_t - \mathbf{z}_t,\end{aligned} \tag{37}$$

Collectively, (36) and (37) imply

$$\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1} = (\mathbf{x}_t - \tilde{\mathbf{x}}_t) + (\gamma\mathbf{g}_t - \Delta_t). \tag{38}$$

Since $\mathbf{y}_0 = \tilde{\mathbf{y}}_0$ using induction yields

$$\begin{aligned}\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1} &= \sum_{j=0}^{t}(\gamma\mathbf{g}_t - \Delta_j) \\ &= \sum_{j=0}^{t}(\mathbf{m}_{j+1} - \mathbf{m}_j) \\ &= \mathbf{m}_{t+1} - \mathbf{m}_0 = \mathbf{m}_{t+1},\end{aligned} \tag{39}$$

where we used the fact that $\mathbf{m}_0 = \mathbf{0}$. ■

G.4 Proof of Lemma 6

Proof. Recall that $\mathbf{g}_t = \frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}} \nabla f_i(\mathbf{x}_t, z_t^i)$, i.e., the average of stochastic gradients ove uncorrupted samples at time t, and $\mathbb{E}[\mathbf{g}_t] = \bar{\mathbf{g}}_t$. By the definition of L-smoothness (see Assumption 2), we have

$$\begin{aligned}f(\tilde{\mathbf{x}}_{t+1}) &\leq f(\tilde{\mathbf{x}}_t) + \langle \nabla f(\tilde{\mathbf{x}}_t), \tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t\rangle + \frac{L}{2}\|\tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t\|^2 \\ &= f(\tilde{\mathbf{x}}_t) + \langle \nabla f(\tilde{\mathbf{x}}_t), -\gamma\mathbf{g}_t - \mathbf{z}_t\rangle + \frac{L}{2}\|\gamma\mathbf{g}_t + \mathbf{z}_t\|^2 \\ &\leq f(\tilde{\mathbf{x}}_t) - \gamma\langle \nabla f(\tilde{\mathbf{x}}_t), \mathbf{g}_t\rangle - \langle \nabla f(\tilde{\mathbf{x}}_t), \mathbf{z}_t\rangle + L\gamma^2\|\mathbf{g}_t\|^2 + L\|\mathbf{z}_t\|^2.\end{aligned} \tag{40}$$

Let $\mathbb{E}_t$ denote expectation with respect to sources of randomness in computation of stochastic gradients at time t. Then,

$$\mathbb{E}_t[f(\tilde{\mathbf{x}}_{t+1})] \leq f(\tilde{\mathbf{x}}_t) - \gamma\langle \nabla f(\tilde{\mathbf{x}}_t), \bar{\mathbf{g}}_t\rangle - \mathbb{E}_t[\langle \nabla f(\tilde{\mathbf{x}}_t), \mathbf{z}_t\rangle] + L\gamma^2\mathbb{E}_t\|\mathbf{g}_t\|^2 + L\mathbb{E}_t\|\mathbf{z}_t\|^2. \tag{41}$$

The first inner-product in (41) can be bounded according to

$$\begin{aligned}
2\langle \nabla f(\tilde{\mathbf{x}}_t), \bar{\mathbf{g}}_t\rangle &= \|\nabla f(\tilde{\mathbf{x}}_t)\|^2 + \|\bar{\mathbf{g}}_t\|^2 - \|\nabla f(\tilde{\mathbf{x}}_t) - \bar{\mathbf{g}}_t\|^2 \\
&\geq \|\nabla f(\tilde{\mathbf{x}}_t)\|^2 - \|\nabla f(\tilde{\mathbf{x}}_t) - \bar{\mathbf{g}}_t\|^2 \\
&= \|\nabla f(\tilde{\mathbf{x}}_t)\|^2 - \|\frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}} \nabla f_i(\tilde{\mathbf{x}}_t) - \frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}} \nabla f_i(\mathbf{x}_t)\|^2 \\
&\geq \|\nabla f(\tilde{\mathbf{x}}_t)\|^2 - \frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}} \|\nabla f_i(\tilde{\mathbf{x}}_t) - \nabla f_i(\mathbf{x}_t)\|^2 \\
&\geq \|\nabla f(\tilde{\mathbf{x}}_t)\|^2 - L^2\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2 \\
&= \|\nabla f(\tilde{\mathbf{x}}_t)\|^2 + L^2\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2 - 2L^2\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2 \\
&\geq \|\nabla f(\tilde{\mathbf{x}}_t)\|^2 + \|\nabla f(\tilde{\mathbf{x}}_t) - \nabla f(\mathbf{x}_t)\|^2 - 2L^2\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2 \\
&= \|\nabla f(\tilde{\mathbf{x}}_t)\|^2 + \|\nabla f(\tilde{\mathbf{x}}_t) - \nabla f(\mathbf{x}_t)\|^2 - 2L^2\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2 \\
&\geq \frac{1}{2}\|\nabla f(\mathbf{x}_t)\|^2 - 2L^2\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2,
\end{aligned} \tag{42}$$

where we employed Fact 1 and L-smoothness of each function several times. Therefore,

$$-\gamma\langle \nabla f(\tilde{\mathbf{x}}_t), \bar{\mathbf{g}}_t\rangle \leq -\frac{\gamma}{4}\|\nabla f(\mathbf{x}_t)\|^2 + \gamma L^2\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2. \tag{43}$$

We now bound the second inner-product. To this end,

$$\begin{aligned}
-\mathbb{E}_t[\langle \nabla f(\tilde{\mathbf{x}}_t), \mathbf{z}_t\rangle] &= -\mathbb{E}_t[\langle \nabla f(\mathbf{x}_t), \mathbf{z}_t\rangle] + \mathbb{E}_t[\langle \nabla f(\mathbf{x}_t) - \nabla f(\tilde{\mathbf{x}}_t), \mathbf{z}_t\rangle] \\
&\leq \rho\gamma\|\nabla f(\mathbf{x}_t)\|^2 + \frac{1}{2\rho\gamma}\mathbb{E}_t\|\mathbf{z}_t\|^2 + \mathbb{E}_t[\langle \nabla f(\mathbf{x}_t) - \nabla f(\tilde{\mathbf{x}}_t), \mathbf{z}_t\rangle] \\
&\leq \frac{\rho\gamma}{2}\|\nabla f(\mathbf{x}_t)\|^2 + \frac{1}{2\rho\gamma}\mathbb{E}_t\|\mathbf{z}_t\|^2 + \frac{1}{2\gamma}\mathbb{E}_t\|\mathbf{z}_t\|^2 + \frac{\gamma}{2}\|\nabla f(\mathbf{x}_t) - \nabla f(\tilde{\mathbf{x}}_t)\|^2 \\
&\leq \frac{\rho\gamma}{2}\|\nabla f(\mathbf{x}_t)\|^2 + \left(\frac{1}{2\rho\gamma} + \frac{1}{2\gamma}\right)\mathbb{E}_t\|\mathbf{z}_t\|^2 + \frac{\gamma L^2}{2}\|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|^2,
\end{aligned} \tag{44}$$

where we used Fact 2 twice and to obtain the last inequality we employed the smoothness assumption. Here, $0 < \rho < 0.5$ is a parameter whose value will be determined later.

Application of (43) and (44) in (41) yields

$$\begin{aligned}
\mathbb{E}_t[f(\tilde{\mathbf{x}}_{t+1})] \leq f(\tilde{\mathbf{x}}_t) &- \left(\frac{1}{2} - \rho\right)\frac{\gamma}{2}\|\nabla f(\mathbf{x}_t)\|^2 + \frac{3\gamma L^2}{2}\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2 \\
&+ L\gamma^2\mathbb{E}_t\|\mathbf{g}_t\|^2 + \left(L + \frac{1}{2\rho\gamma} + \frac{1}{2\gamma}\right)\mathbb{E}_t\|\mathbf{z}_t\|^2.
\end{aligned} \tag{45}$$

■

H Proof of Theorem 1

Proof. Rearranging (45) and taking expectation with respect to the entire sources of randomness, i.e. the proposed coordinated sparse approximation and the randomness in computation of stochastic gradient in iterations $0, \ldots, t-1$, using the bounded SFO assumption yields

$$\begin{aligned}
\left(\frac{1}{2} - \rho\right)\frac{\gamma}{2}\mathbb{E}\|\nabla f(\mathbf{x}_t)\|^2 \leq &\mathbb{E}[f(\tilde{\mathbf{x}}_t)] - \mathbb{E}[f(\tilde{\mathbf{x}}_{t+1})] + \frac{3\gamma L^2}{2}\mathbb{E}\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2 \\
&+ L\gamma^2\sigma^2 + \left(L + \frac{1}{2\rho\gamma} + \frac{1}{2\gamma}\right)\mathbb{E}\|\mathbf{z}_t\|^2.
\end{aligned} \tag{46}$$

Evidently, we need to bound two quantities in (46). Lemma 4 establishes a bound on $\mathbb{E}\|\mathbf{z}_t\|^2$ while by Lemma 5

$$\mathbb{E}\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2 = \mathbb{E}\|\mathbf{m}_t\|^2 \leq \frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}}\mathbb{E}\|\hat{\mathbf{m}}_t\|^2 \leq \frac{4(1-\xi^2)\gamma^2\sigma^2}{\xi^2} \tag{47}$$

using Lemma 3 we get :

$$\begin{aligned}\left(\frac{1}{2}-\rho\right)\frac{\gamma}{2}\mathbb{E}\|\nabla f(\mathbf{x}_t)\|^2 \leq & \mathbb{E}[f(\tilde{\mathbf{x}}_t)] - \mathbb{E}[f(\tilde{\mathbf{x}}_{t+1})] + L\gamma^2\sigma^2 \\ & + \frac{6\gamma L^2(1-\xi^2)\gamma^2\sigma^2}{\xi^2} \\ & + \left(L+\frac{1}{2\rho\gamma}+\frac{1}{2\gamma}\right)\frac{96\gamma^2\sigma^2}{(1-\alpha)^2}\left[1+\frac{4(1-\xi^2)}{\xi^2}\right] \\ & + \left(L+\frac{1}{2\rho\gamma}+\frac{1}{2\gamma}\right)\frac{2\epsilon^2}{|\mathbb{G}|^2(1-\alpha)^2}.\end{aligned} \tag{48}$$

Finally, letting $\rho = 1/4$ and $\gamma = \frac{1}{2L}$ we obtain the stated result by averaging (48) over time and noting $f^* \leq \mathbb{E}[f(\tilde{\mathbf{x}}_T)]$ and $f(\tilde{\mathbf{x}}_0) = f(\mathbf{x}_0)$. ■

I Proof of Theorem 2

Recall (46) in the proof of Theorem 1. Let $a_t := \mathbb{E}[f(\tilde{\mathbf{x}}_t)] - f^* \geq 0$. Using Assumption 3 to bound the gradient terms in (46) yields

$$\begin{aligned} a_{t+1} \leq & a_t - \left(\frac{1}{2}-\rho\right)\gamma\mu\mathbb{E}[f(\mathbf{x}_t)-f^*] + \frac{3\gamma L^2}{2}\mathbb{E}\|\tilde{\mathbf{x}}_t-\mathbf{x}_t\|^2 \\ & + L\gamma^2\sigma^2 + \left(L+\frac{1}{2\rho\gamma}+\frac{1}{2\gamma}\right)\mathbb{E}\|\mathbf{z}_t\|^2. \end{aligned} \tag{49}$$

The above result is not sufficient to complete the proof since a_t is with respect to the virtual sequence. This difficulty, which is addressed for the first time in this paper, has been the main challenge in the proof of convergence of error compensated schemes with biased gradient compression under PLC. To deal with this burden, we revisit (41) to establish an alternative bound. Specifically, we to bound the first inner-product in (41) we instead establish

$$\begin{aligned} -\gamma\langle\nabla f(\tilde{\mathbf{x}}_t), \bar{\mathbf{g}}_t\rangle &= -\gamma\langle\nabla f(\tilde{\mathbf{x}}_t), \nabla f(\tilde{\mathbf{x}}_t)\rangle + \gamma\langle\nabla f(\tilde{\mathbf{x}}_t), \nabla f(\tilde{\mathbf{x}}_t)-\bar{\mathbf{g}}_t\rangle \\ &\leq -\gamma\|\nabla f(\tilde{\mathbf{x}}_t)\|^2 + \frac{\gamma\rho_1}{2}\|\nabla f(\tilde{\mathbf{x}}_t)\|^2 + \frac{\gamma}{2\rho}\|\nabla f(\tilde{\mathbf{x}}_t)-\bar{\mathbf{g}}_t\|^2 \\ &= -\gamma(1-\frac{\rho_1}{2})\|\nabla f(\tilde{\mathbf{x}}_t)\|^2 + \frac{\gamma}{2\rho_1}\|\nabla f(\tilde{\mathbf{x}}_t)-\bar{\mathbf{g}}_t\|^2 \end{aligned} \tag{50}$$

where we used Fact 2 with parameter $\rho_1 > 0$ which will be determined later. Recall that by smoothness

$$\|\nabla f(\tilde{\mathbf{x}}_t)-\bar{\mathbf{g}}_t\|^2 \leq L^2\|\tilde{\mathbf{x}}_t-\mathbf{x}_t\|^2. \tag{51}$$

We now derive a bound for the second inner-product in (41). To do so, an application of Fact 2 yields

$$-\mathbb{E}_t[\langle\nabla f(\tilde{\mathbf{x}}_t), \mathbf{z}_t\rangle] \leq \frac{\rho_2\gamma}{2}\|\nabla f(\tilde{\mathbf{x}}_t)\|^2 + \frac{1}{2\gamma\rho_2}\mathbb{E}_t\|\mathbf{z}_t\|^2. \tag{52}$$

Therefore, in light of these new bounds we obtain

$$\begin{aligned} \mathbb{E}_t[f(\tilde{\mathbf{x}}_{t+1})] \leq f(\tilde{\mathbf{x}}_t) - \gamma\left(1-\frac{\rho_1+\rho_2}{2}\right)\|\nabla f(\tilde{\mathbf{x}}_t)\|^2 + L\gamma^2\mathbb{E}_t\|\mathbf{g}_t\|^2 \\ + \left(L+\frac{1}{2\gamma\rho_2}\right)\mathbb{E}_t\|\mathbf{z}_t\|^2 + \frac{\gamma L^2}{2\rho_1}\|\tilde{\mathbf{x}}_t-\mathbf{x}_t\|^2. \end{aligned} \tag{53}$$

Subtracting f^* and taking expectation with respect to the entire sources of randomness yields

$$\begin{aligned} \mathbb{E}[f(\tilde{\mathbf{x}}_{t+1})] - f^* \leq \mathbb{E}[f(\tilde{\mathbf{x}}_t)] - f^* - \gamma\left(1-\frac{\rho_1+\rho_2}{2}\right)\mathbb{E}\|\nabla f(\tilde{\mathbf{x}}_t)\|^2 + L\gamma^2\sigma^2 \\ + \left(L+\frac{1}{2\gamma\rho_2}\right)\mathbb{E}\|\mathbf{z}_t\|^2 + \frac{\gamma L^2}{2\rho_1}\mathbb{E}\|\tilde{\mathbf{x}}_t-\mathbf{x}_t\|^2. \end{aligned} \tag{54}$$

The gradient term $\mathbb{E}\|\nabla f(\tilde{\mathbf{x}}_t)\|^2$ in (54) can be related to a_t using the PL condition. Therefore

$$\begin{aligned} a_{t+1} \leq & \left[1 - 2\mu\gamma\left(1 - \frac{\rho_1 + \rho_2}{2}\right)\right] a_t + L\gamma^2\sigma^2 \\ & + \left(L + \frac{1}{2\gamma\rho_2}\right)\mathbb{E}\|\mathbf{z}_t\|^2 + \frac{\gamma L^2}{2\rho_1}\mathbb{E}\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2. \end{aligned} \tag{55}$$

Multiply (49) and (55) by 1/2 and add the two to obtain

$$\begin{aligned} a_{t+1} \leq & \left[1 - \mu\gamma\left(1 - \frac{\rho_1 + \rho_2}{2}\right)\right] a_t + L\gamma^2\sigma^2 - \left(\frac{1}{2} - \rho\right)\frac{\gamma\mu}{2}\mathbb{E}[f(\mathbf{x}_t) - f^*] \\ & + \left(L + \frac{1}{4\gamma\rho} + \frac{1}{4\gamma\rho_2} + \frac{1}{4\gamma}\right)\mathbb{E}\|\mathbf{z}_t\|^2 \\ & + \left(3 + \frac{1}{\rho_1}\right)\frac{\gamma L^2}{4}\mathbb{E}\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2. \end{aligned} \tag{56}$$

Define

$$b_t := \mathbb{E}[f(\mathbf{x}_t) - f^*]. \tag{57}$$

Let $\rho = 1/4$, $\rho_1 = 1/2$, $\rho_2 = 1/2$, and $\gamma = 1/4L$. Using Lemma 4 and (47), (56) simplifies to

$$\begin{aligned} a_{t+1} \leq & \left[1 - \frac{\mu\gamma}{2}\right] a_t - \frac{\gamma\mu}{8} b_t + L\gamma^2\sigma^2 + \frac{5L^2}{4}\gamma^3\sigma^2\left[\frac{4(1-\xi^2)}{\xi^2}\right] \\ & + \frac{192\gamma\sigma^2}{(1-\alpha)^2}\left[1 + \frac{4(1-\xi^2)}{\xi^2}\right] + \frac{4\epsilon^2}{\gamma|\mathbb{G}|^2(1-\alpha)^2}. \end{aligned} \tag{58}$$

Thus we can apply Fact 3 to obtain

$$\begin{aligned} \frac{1}{W_T}\sum_{t=0}^{T-1} b_t w_t \leq & \frac{8(f(\mathbf{x}_0) - f^*)}{\mu\gamma}\left[1 - \frac{\mu\gamma}{2}\right]^T + \frac{8L\gamma\sigma^2}{\mu} + \frac{10L^2}{\mu}\gamma^2\sigma^2\left[\frac{4(1-\xi^2)}{\xi^2}\right] \\ & + \frac{1536\sigma^2}{\mu(1-\alpha)^2}\left[1 + \frac{4(1-\xi^2)}{\xi^2}\right] + \frac{32\epsilon^2}{\mu\gamma^2|\mathbb{G}|^2(1-\alpha)^2}, \end{aligned} \tag{59}$$

where we used $E[f(\tilde{\mathbf{x}}_0^i)] = f(\mathbf{x}_0)$. Finally, to obtain the stated result we invoke Fact 4 and the assumption that the solution set (i.e., the set of all stationary points) is convex. Specifically, by the quadratic growth condition and convexity of the Euclidean norm

$$\begin{aligned} \sum_{t=0}^{T-1}\frac{w_t}{W_T}\mathbb{E}[f(\mathbf{x}_t) - f^*] \geq & \sum_{t=0}^{T-1}\frac{w_t}{W_T}\mathbb{E}\|\mathbf{x}_t - \mathcal{P}_{\mathcal{X}^*}(\mathbf{x}_t)\|^2 \\ \geq & \frac{\mu}{2}\sum_{t=0}^{T-1}\frac{w_t}{W_T}\mathbb{E}\|\mathbf{x}_t - \mathcal{P}_{\mathcal{X}^*}(\mathbf{x}_t)\|^2 \\ \geq & \frac{\mu}{2}\mathbb{E}\|\sum_{t=0}^{T-1}\frac{w_t}{W_T}\mathbf{x}_t - \sum_{t=0}^{T-1}\frac{w_t}{W_T}\mathcal{P}_{\mathcal{X}^*}(\mathbf{x}_t)\|^2 \\ := & \frac{\mu}{2}\mathbb{E}\|\hat{\mathbf{x}}_T - \mathbf{x}^*\|^2, \end{aligned} \tag{60}$$

where $\mathcal{P}_{\mathcal{X}^*}(\mathbf{x}_t)$ is the projection of $\mathbf{x}_t$ onto the solution set $\mathcal{X}^*$ and $\mathbf{x}^* := \sum_{t=0}^{T-1}\frac{w_t}{W_T}\mathcal{P}_{\mathcal{X}^*}(\mathbf{x}_t) \in \mathcal{X}^*$ by the assumption that the solution set is convex.