Mitigating Model Poisoning Attacks on Distributed Learning with Heterogeneous Data

Jian Xu[1], Guangfeng Yan[2], Ziyan Zheng[1], Shao-Lun Huang[1*]
[1]Tsinghua Shenzhen International Graduate School, Tsinghua University, [2]City University of Hong Kong

2023 International Conference on Machine Learning and Applications (ICMLA) | 979-8-3503-4534-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICMLA58977.2023.00108

***Abstract*—Gradient-based distributed learning techniques have been essential for machine model training on distributed samples without collecting raw data. However, such learning systems are vulnerable to both internal failures and external attacks. In this work, we study the Byzantine robustness of distributed learning over heterogeneous data for classification tasks, where each worker only contains data samples belonging to some specific classes (aka., label skew) and a fraction of workers are corrupted by a Byzantine adversary to conduct attacks by sending malicious gradients. Existing defenses usually fail under such heterogeneous cases. To remedy this, we propose a gradient decomposition scheme called *DeSGD* to achieve more robust distributed model training. The key idea for mitigating the impact of data heterogeneity on the Byzantine robustness is to divide the full global gradient into individual gradients of each data class and conduct resilient aggregation in a class-wise manner. The proposed framework can easily integrate existing advanced defense methods and local momentum mechanism. Evaluation results on the Fashion-MNIST dataset with various strong attacks demonstrate the improved robustness of learning over distributed data in the presence of both label skew and attacks.**

***Index Terms*—Distributed Optimization, Byzantine Attack, Non-IID Data**

I. Introduction

The remarkable progress of machine learning-based applications can be attributed to increasingly large data volumes and distributed computation that significantly benefits model learning [1]. In real-world applications, due to isolated data storage, privacy concerns and constrained communication resource, collecting data and performing centralized training may be not practical. The gradient-based learning algorithms [2] could be implemented in a distributed fashion for training large-scale deep learning models on scattered data, e.g., deep neural networks for human face identification and speech recognition [3], [4]. Unfortunately, in such learning systems, the server cannot control the behavior of workers and thus is vulnerable to various kinds of system failures and malicious attacks. In particular, the Byzantine attack is viewed as the worst-case scenario, where a Byzantine adversary could corrupt or inject a number of workers to perform attacks by sending malicious gradient messages [5], [6]. Depending on the specific threat assumptions, the Byzantine nodes may have either access to the training data or knowledge of honest gradients, and are even capable of collusion. The primary goal of Byzantine attacks is to degrade the model performance or even prevent successful training [7]. It has been shown that detecting and mitigating Byzantine attacks is crucial for the model safety and utility. Moreover, the decentralized training data may not be independent and identically distributed (IID), which makes it more difficult to detect abnormal gradients as the deviation among honest gradients is also large. Therefore, a robust distributed learning algorithm that is resilient to Byzantine attacks and can handle non-IID data is necessary to fully capitalize on the advantages of distributed learning.

* Corresponding Author: shaolun.huang@sz.tsinghua.edu.cn

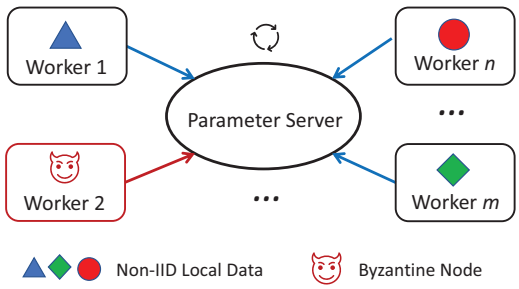


Fig. 1: Distributed learning with Byzantine nodes and non-IID data, where each worker has data from only a few categories.

In recent years, significant efforts have been devoted to developing Byzantine-robust optimization algorithms to mitigate strong attacks and simultaneously ensure the high model performance. A number of works propose statistic-based robust gradient aggregation rules to directly filer the malicious gradients or result in resilient gradient averaging [5], [8], [9]. Specially, if auxiliary validation data is available on the server side, validation-based or gradient reference-based approaches are promising to detect malicious attacks [7], [10], [11]. Nevertheless, existing statistic-based methods are usually based on the IID assumption and are vulnerable to some well-crafted attacks [12]. The local momentum is first proposed in [13] to reduce the gradient variance for improved resilient aggregation. Though the performance-based methods would be generally reliable, they rely on the availability of representative validation data, and the server must evaluate all received gradients individually, thus the computation costs increase linearly with system scale. Compared to the IID settings, research on robust and efficient algorithms against Byzantine attacks in non-IID settings is far from satisfactory. Combining node clustering and outlier-robust distributed optimization to learn multiple global models is proposed in [14]. A bucketing algorithm is applied in [15] to make the input vectors before aggregation more homogeneous.

In this work, we mainly focus on the typical non-IID scenarios induced by label skew and propose a training scheme that can be combined with existing defenses. Inspired by the general convergence of SGD in centralized learning systems, where even stochastic gradient on a single data point can lead

1946-0759/23/$31.00 ©2023 IEEE
DOI 10.1109/ICMLA58977.2023.00108

to model convergence, we develop a novel and resilient algorithm for non-IID distributed learning by computing class-wise gradient and incorporating the existing defenses developed for IID systems. To the best of our knowledge, this is the first work to employ decomposed aggregation of class-wise gradient for mitigating Byzantine attacks in distributed systems. Our developed algorithm tries to mimic the IID scenarios and then employ off-the-shelf robust aggregation rules. The proposed framework is evaluated on the benchmark dataset and the numerical results demonstrate its effectiveness in safeguarding the distributed model training on non-IID data.

II. Problem Setup

A. Threat Model

As illustrated in Fig. 1, we focus on server-based distributed learning and consider the existence of a Byzantine adversary. The adversary corrupts f nodes among the n workers (denoted by $\mathcal{B}$) to perform model poisoning attacks. Specifically, we assume the adversary has knowledge of all gradients from honest workers and could collude to perform strong attacks, but the adversary could not corrupt the server. The number of corrupted workers is unknown to the server. The tolerable upper bound of f may depend on the label distributions of honest workers, which will be discussed in section III-C.

B. Problem Formulation

We focus on distributed learning in label skewed settings and assume that each worker only has access to a few classes of data while the label-conditional data distribution is the same across workers. All worker nodes will jointly train a shared model based on disjoint local data. Mathematically, the underlying optimization problem can be formalized as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\xi_i\sim D_i}\left[F(\mathbf{x};\xi_i)\right], \tag{1}$$

where n is the total number of workers, D_i denotes the local dataset of worker *i*, and $F(\mathbf{x};\xi_i)$ denotes the local loss function given shared model parameters $\mathbf{x}\in\mathbb{R}^d$ and the training data ξ_i. In the traditional framework, at each iteration, the *i*-th honest worker randomly draws a mini-batch data from D_i and computes a local stochastic gradient with respect to the global shared parameter $\mathbf{x}$, while the Byzantine worker can send arbitrary gradient message:

$$g_t^{(i)} = \begin{cases} \nabla F(\mathbf{x}_t;\xi_i), & \text{if worker } i \text{ is honest} \\ \text{arbitrary}, & \text{if worker } i \text{ is Byzantine} \end{cases} \tag{2}$$

The parameter server then collects the local gradients and employs a robust gradient aggregation rule to obtain a global gradient to update the model parameters as follows:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \cdot Agg(\{g_t^{(i)}\}_{i=1}^n). \tag{3}$$

Then, the result will be broadcast to all worker nodes to update their local models and start a new iteration. This framework works well under IID and less non-IID settings, but it is still vulnerable in highly non-IID scenarios, e.g., when each worker node contains training samples from only a subset of classes.

III. The Proposed Framework

In this section, we introduce our gradient decomposition framework and practical algorithm design for robust distributed learning step-by-step, where we apply class-wise gradient (momentum) collection and conduct aggregation with off-the-shelf defenses.

A. Class-wise Gradient

As stated in Section II-B, traditional distributed learning algorithms compute the local gradient based on randomly sampled data points. When local label distributions are highly skewed, those approaches result in high variance across workers. Therefore, our simple yet effective strategy is to mimic an IID setting during the aggregation by computing class-wise gradients (class-gradients). These gradients are on data samples belonging to a specific class $c\in[C]$:

$$g_t^{(c,i)} = \nabla F(\mathbf{x}_t;\xi_{i,c}), \tag{4}$$

where i refers to the index of any worker that has samples of class c, and $\xi_{i,c}$ denotes the sampled data points. It is worth noting that in our label-skew based non-IID settings, not all honest workers have data from all classes, but the Byzantine workers can always choose to compute either the true class-gradient or a malicious gradient. Additionally, instead of a single class, the server can also request the workers to send a set S_t of class-gradients at round t with a certain scheduling rule for the set selection. Consider that the per-round communication overhead is proportional to the size $K=|S_t|$, a small set might be preferred in practice.

B. Robust Gradient Aggregation

The robust aggregation rules aim to mitigate the Byzantine attacks by resilient averaging of collected gradients. The idea behind resilient averaging is to ensure that the distance between the aggregation result and the average of honest gradients is bounded by their diameter times a factor λ. The formal definition is stated below [16].

Definition 1 (**(f,λ)-Resilient averaging**). For any collection of gradient vectors $\{g^{(i)}\}_{i=1}^n$ generated by $n-f$ honest workers (denoted by $\mathcal{G}$) and f Byzantine clients, an aggregation rule is called (f,λ)-Resilient averaging if there exists a positive constant λ such that the output $\hat{g}$ satisfies:

$$\|\hat{g}-\bar{g}\|^2 \le \lambda \sup_{i,j\in\mathcal{G}} \mathbb{E}[\|g^{(i)}-g^{(j)}\|^2], \tag{5}$$

where $\bar{g}=\frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}} g^{(i)}$ is the average of honest gradients.

The popular defenses, such as Krum and Geometric Median, are belonging to such resilient averaging. Essentially, a smaller λ indicates better resilience. Let C_i denote the class set at worker i and n_c denote the set of workers that have samples from class c. At each iteration t, the server can randomly select a class set S_t for class-gradient collection and aggregation:

$$\hat{g}_t^c = Agg(\{g_t^{(c,i)}\}_{i\in[n_c]}), \forall c\in S_t. \tag{6}$$

Then the global model will be updated by the following gradient decent step with a learning rate η:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta\frac{1}{|S_t|}\sum_{c\in S_t}\hat{g}_t^c. \tag{7}$$

C. Convergence Analysis and Insights

In this part, we demonstrate the convergence guarantee of non-convex optimization problems under Byzantine attacks. To conduct the convergence analysis, we make the following assumptions that are commonly used in the literature [14], [16], [17] for theoretical analysis of distributed optimization.

Assumption 1. *The problem formulated by (1) satisfies:*

1. ***Smoothness****: The objective function $F(\cdot)$ is smooth with Lipschitz constant $L > 0$, which implies that:*

$$F(\mathbf{x}) - F(\mathbf{y}) \leq \nabla F(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) + \frac{L}{2}\|\mathbf{x}-\mathbf{y}\|^2. \quad (8)$$

2. ***Unbiased gradient****: For each honest worker, the local stochastic gradient of the class-c is unbiased:*

$$\mathbb{E}_{\xi\sim D_c}[\nabla F(\mathbf{x};\xi)] = \nabla F_c(\mathbf{x}). \quad (9)$$

3. ***Bounded variance****: The class-wise gradient of each honest worker has a bounded intra-class variance and inter-class variance:*

$$\mathbb{E}_{\xi\sim D_c}\left[\|\nabla F(\mathbf{x};\xi) - \nabla F_c(\mathbf{x})\|^2\right] \leq \sigma^2, \quad (10)$$

$$[\|\nabla F_c(\mathbf{x}) - \nabla F(\mathbf{x})\|^2] \leq \zeta^2. \quad (11)$$

Assumption 2. *There are n_c honest workers containing samples belonging to class-c for each class $c \in [C]$ and n_c is larger than the maximum number of Byzantine nodes, which means that there exists a positive m such that $f < m \leq n_c$.*

Remark 1. *Assumption 1 is actually equivalent to the bounded intra-node gradient variance and the bounded deviation between local and global gradients. Assumption 2 means that for each class we can always collect more honest gradients than the malicious ones, which is essentially required to generate a global model with good performances on all classes.*

Assumption 3. *The class set selection S_t is unbiased, i.e, we have $\mathbb{E}[\frac{1}{|S_t|}\sum_{c\in S_t}\bar{g}_t^c] = \nabla F(\mathbf{x}_t)$ over the training process.*

Based on the above assumptions, we can characterize the convergence property of our method by the following theorem.

Theorem 1. *For problem (1) under Assumption 1-2, suppose the (b, λ)-Resilient defense and DeSGD are employed with a fixed learning rate $\eta \leq 1/(4L)$, $F^* = \min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x})$, then we have the following convergence result:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\mathbf{x}_t)\|^2] \leq \frac{4[F(\mathbf{x}_0)-F^*]}{\eta T} + 4\eta L\Delta_1 + \Delta_2, \quad (12)$$

where $\Delta_1 = \frac{(C-K)\zeta^2}{(C-1)K} + (2\lambda + \frac{1}{m})\sigma^2$ and $\Delta_2 = 4\lambda\sigma^2$.

Proof. For any class c, by taking the expectation on local sampling, we obtain the bounded class-wise gradient deviation as

$$\mathbb{E}\left\|\hat{g}_t^{(c)} - \nabla F_c(\mathbf{x}_t)\right\|^2 = \mathbb{E}\left\|\hat{g}_t^{(c)} - \bar{g}_t^{(c)} + \bar{g}_t^{(c)} - \nabla F_c(\mathbf{x}_t)\right\|^2$$
$$= \mathbb{E}\left\|\hat{g}_t^{(c)} - \bar{g}_t^{(c)}\right\|^2 + \mathbb{E}\left\|\bar{g}_t^{(c)} - \nabla F_c(\mathbf{x}_t)\right\|^2 \leq (2\lambda + \frac{1}{m})\sigma^2.$$

Taking the expectation on class set selection, according to the Lemma 4 in [18], we have

$$\mathbb{E}\left\|\frac{1}{K}\sum_{c\in S_t}\nabla F_c(\mathbf{x}_t) - \nabla F(\mathbf{x}_t)\right\|^2 \leq \frac{(C-K)\zeta^2}{(C-1)K}.$$

Combing the above results, we can bound the deviation between the estimated $\hat{g}_t$ and the true gradient $\nabla F(\mathbf{x}_t)$ by

$$\mathbb{E}\|\hat{g}_t - \nabla F(\mathbf{x}_t)\|^2 = \mathbb{E}\left\|\frac{1}{K}\sum_{c\in S_t}[\hat{g}_t^{(c)} - \nabla F_c(\mathbf{x}_t)] + \right.$$
$$\left.\frac{1}{K}\sum_{c\in S_t}[\nabla F_c(\mathbf{x}_t) - \nabla F(\mathbf{x}_t)]\right\|^2 \leq \frac{(C-K)\zeta^2}{(C-1)K} + (2\lambda + \frac{1}{m})\sigma^2$$

Then, due to the smoothness of objective function F and $\eta \leq 1/(4L)$, at iteration t we have

$$\mathbb{E}[F(\mathbf{x}_{t+1})] - F(\mathbf{x}_t) \leq -\eta\langle\nabla F(\mathbf{x}_t), \mathbb{E}[\hat{g}_t]\rangle + \frac{L\eta^2}{2}\mathbb{E}\|\hat{g}_t\|^2$$
$$= -\eta\|\nabla F(\mathbf{x}_t)\|\|\mathbb{E}[\hat{g}_t - \bar{g}_t]\| - \eta\|\nabla F(\mathbf{x}_t)\|^2 + \frac{L\eta^2}{2}\mathbb{E}\|\hat{g}_t\|^2$$
$$\leq -\frac{\eta}{4}\|\nabla F(\mathbf{x}_t)\|^2 + \eta\lambda\sigma^2 + L\eta^2\mathbb{E}\left[\|\hat{g}_t - \nabla F(\mathbf{x}_t)\|^2\right]$$

Taking the total expectation and rearranging the terms, we get

$$\frac{\eta}{4}\mathbb{E}[\|\nabla F(\mathbf{x}_t)\|^2] \leq \mathbb{E}[F(\mathbf{x}_t) - F(\mathbf{x}_{t+1})]$$
$$+ \eta\lambda\sigma^2 + L\eta^2\left[\frac{(C-K)\zeta^2}{(C-1)K} + (2\lambda + \frac{1}{m})\sigma^2\right]$$

Taking summation and dividing by $\eta T/4$ gives the final result. □

Remark 2. *The result in Theorem 1 shows that when $K < C$, DeSGD will decrease the convergence rate and increase the convergence error. However, a large K will incur high communication overhead, which means there is a trade-off between effciency and robustness. When no Byzantine adversaries exist and all class-gradients are collected with class-wise mean averaging (thus λ=0), taking $\eta=\sqrt{\frac{m}{T}}$ with a sufficiently large T, we can recover the well-known $O(\frac{1}{\sqrt{mT}})$ convergence rate.*

D. Practical Algorithm Design

Ideally, the proposed framework may request all class-gradients in each iteration for better global gradient estimation, i.e., $S_t := [C]$. When the number of workers or the number of total categories is large, the increased communication overhead might be unacceptable for some realistic scenarios. Additionally, unbiased class set selection is also hard to guarantee in practice. Moreover, while the momentum mechanism could be further employed to accelerate the model convergence in non-Byzantine setting, it may worsen the model training under attacks as the global momentum could accumulate hidden malicious components from past local gradients. To overcome these limitations, we develop two key designs by (i) cyclically choosing one class c_t in each iteration such that each class has approximately the same chance to be selected over the long-term training process without inducing extra communication overhead; (ii) extending the local momentum mechanism [13],

[16] into a class-wise manner and storing them in the local workers, where the local class-momentum can reduce the variance of local-gradient across workers to facilitate the defense[1]. It is worth noting that each honest worker-i will compute all class-momentum for $c \in C_i$ in each iteration, and the computation for $c \in C_i \setminus c_t$ could be conducted during the worker-server communication interval to further improve the system efficiency. For all baseline methods, we also use the local momentum mechanism for fair comparisons.

The overall algorithm is illustrated in Alg.1. In each iteration t, the server collects the local class-momentums and conduct class-wise aggregation by

$$\hat{m}_t^{c_t} = Agg(\{m_t^{(c_t,i)}\}_{i\in[n_{c_t}]}) \tag{13}$$

Specifically, we establish a global class-momentum buffer to store the historical class-momentum to compensate for the missing class-momentums during the server aggregation. We simply average the available class-momentums as (14) and the update rule of momentum buffer is designed as (15), where a decay coefficient β is used to mitigate the effect of staleness of some class-momentums.

$$\bar{m}_t = \frac{1}{|\mathcal{M}|}\sum_{c\in\mathcal{M}} m_g^c \tag{14}$$

$$m_g^c := \begin{cases} \beta m_g^c, & c \neq c_t \\ \hat{m}_t^c, & c = c_t \end{cases} \tag{15}$$

Finally, the global model update can be formulated by combing the new-class momentum and historical class-averaged momentum as in (16), where we set $\gamma = 0.5$ by default.

$$\Delta \mathbf{x}_t = (1-\gamma)\hat{m}_t + \gamma\bar{m}_t \tag{16}$$

IV. Empirical Evaluation

A. Experimental Setup

Datasets and Models: We use the benchmark dataset Fashion-MNIST with a convolution neural network to conduct distributed learning tasks. The dataset is 10-category classification task and we distribute the training samples randomly and evenly from 3 different classes to each of the honest workers to simulate the label skew. The CNN model is constructed by two convolution layers, each followed by a max-pooling layer, and two fully-connected layers.

Training settings: We consider a setup with 20 honest workers and 4 Byzantine nodes, where the Byzantine nodes are assumed to have data samples of all classes. We employ the stochastic gradient decent as the local optimizer with a mini-batch size 50 and a momentum coefficient $\beta = 0.9$. Only one class-momentum is collected in each round and the class c is cyclically selected over the whole class set. Each training algorithm is run for 3000 iterations with a fixed learning rate 0.1 and the weight decay is set to 0.0005. The best achieved test accuracy during the training process is reported to indicate the capability of defense[2].

[1]Hereinafter, we use the class-monentum to replace the class-gradient.

[2]We report the average accuracy over multiple trials.

Algorithm 1 Byz-DeSGD

Input: Learning rate η, iterations T, local momentum β, compensation coefficient γ, aggregation rule $Agg(\cdot)$
Initialize: Initial model $\mathbf{x}_0 \in \mathbb{R}^d$, global buffer $\mathcal{M} = \mathbf{0}$
for $t = 0, 1, ..., T-1$ **do**
  Server:
  Broadcast $\mathbf{x}_t$ and class c_t
  Workers in parallel :
  • Honest:
  - Compute stochastic class-gradient: $g_t^{(c,i)}$ for $c \in C_i$
  - Compute local class-momentum:
    $m_t^{(c,i)} = \beta g_t^{(c,i)} + (1-\beta)m_{t-1}^{(c,i)}$ for $c \in C_i$
  - Send class-momentum(s) $m_t^{(c_t,i)}$ if $c_t \in C_i$
  • Corrupted:
  - Send $\star$ (arbitrary) to the server
  Server:
  Collect class-moments and conduct aggregation: $\hat{m}_t = Agg(\{m_t^{(c_t,i)}\}_{i\in[n_{c_t}]})$
  Get global class-average momentum $\bar{m}_t$ in (14)
  Update model: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta((1-\gamma)\hat{m}_t + \gamma\bar{m}_t)$
  Update global momentum buffer $\mathcal{M}$ in (15)
end for

Evaluated Attacks: We consider the following commonly used model poisoning attacks in the literature.

(i) Constant Attack: The Byzantine worker sends a constant vector as $g_m = c_1 \cdot \mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^d$ is the all-ones vector and we take the $c_1 = 5.0$ in all experiments.

(ii) Gaussian Attack: The element of each coordinate is independently generated by a Gaussian distribution $\mathcal{N}(\mu, \sigma_1^2)$, where we set $\mu = 0$ and $\sigma_1 = 2.0$.

(iii) Sign-Flip Attack: The Byzantine worker first computes local gradient and then reverses it with scaling: $g_m = -c_2 \cdot g$ to conduct attack. We set $c_2 = 1.0$ in all experiments.

(iv) Inner-Product Manipulation (IPM): The Byzantine worker first computes mean gradient of honest workers $\tilde{g}$ and then reverses it with a small scaling: $g_m = -\epsilon \cdot \tilde{g}$. We set $\epsilon = 0.1$ as recommended in the original paper.

(v) A Little is Enough (ALIE): As proposed in [12], the Byzantine workers calculate coordinate-wise mean μ and std σ, and then send malicious gradient vector with elements crafted as: $g_m = \mu - z \cdot \sigma$, where we always set $z = 1.0$.

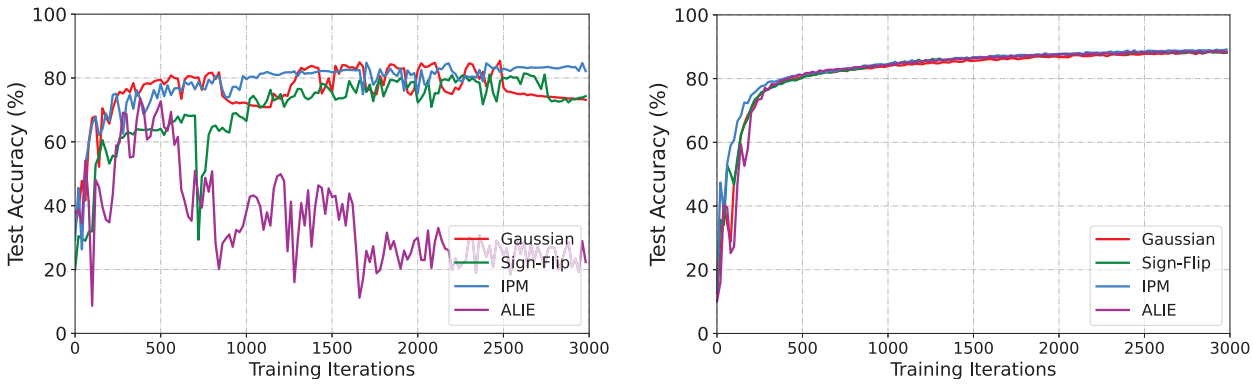
(vi) Mimic Attack: All the Byzantine workers copy the message of one particular honest worker and send to the server to induce a high model bias.

B. Main Results and Analysis

We evaluate the performance of popular defense methods, including Krum/Multi-Krum [5], trimmed-mean (TrMean) and geometric median (GeoMed) [19], FABA [20] DnC [21], Clustering [22] and Centered-Clipping [17]. Note that some methods require the proportion of Byzantine workers. In particular, we evaluate the plain Mean aggregation with clipping (ClipMean) with the clipping bound set to the median of magnitudes. Then, we equip our scheme to several methods.

TABLE I: Highest accuracy (%) during the training process under different attacks on Fashion-MNIST. The mark '+' means *DeSGD* is applied. The distributed SGD with global momentum without any defenses/attacks can reach an accuracy of 90.58%.

Method	**Highest Accuracy under Attacks (%) (↑)**							
	Non	Mimic	Constant	Gaussian	Label-Flip	Sign-Flip	IPM	ALIE
ClipMean	90.38	88.63	74.14	89.92	90.12	88.82	89.67	65.11
Krum	89.18	58.08	63.92	61.35	89.27	42.86	46.25	62.65
GeoMed	89.87	48.21	53.91	53.49	90.02	43.64	46.16	43.29
Clustering	84.81	87.63	67.87	84.91	86.61	81.15	83.58	73.63
CenteredClip	89.12	87.42	84.12	76.53	90.12	83.47	87.54	72.23
M-Krum	89.62	82.56	89.92	89.68	89.14	41.51	66.65	64.24
TrMean	87.45	85.59	73.64	88.73	84.14	79.58	85.45	68.21
FABA	89.74	89.23	89.91	90.12	89.81	87.29	88.89	75.24
DnC	90.52	88.53	89.96	88.49	90.06	87.07	88.79	76.94
Krum+	88.33	86.39	88.79	88.94	89.15	88.73	87.74	69.07
M-Krum+	88.25	87.14	89.09	88.97	88.59	88.67	88.53	78.48
GeoMed+	89.73	86.23	88.85	88.49	89.02	88.18	88.68	75.87
TrMean+	88.75	87.95	78.28	89.10	88.76	86.78	87.43	80.44
DnC+	89.75	88.85	89.03	88.98	89.17	88.13	88.42	85.36
Clustering+	88.79	88.46	88.41	88.39	89.43	88.52	88.96	88.66



(a) Vanilla SGD with local momentum

(b) DeSGD with local momentum

Fig. 2: Test accuracy over iterations with Clustering defense and various attacks. Left: traditional distributed training local momentum. Right: DeSGD with local momentum. The training stability is significantly improved under the DeSGD scheme.

Performance Comparison. We test the above defense methods under different attacks and the main results of best achieved model accuracy are reported in Table I, from which we can find that under the common distributed learning framework, all defense methods will result in relatively poor model accuracy under specific attacks. For example, Krum and GeoMed have relatively poor performance as they usually result in highly biased gradient aggregate in non-IID settings, TrMean and Multi-Krum have more accuracy drops under sign-flip attack while Clustering is less robust to constant attack. When our proposed scheme is employed with some defenses, the robustness is generally improved, e.g., DnC and Clustetring with DeSGD perform well under various attacks. Note that the ALIE is originally a strong attack in the IID settings and most existing defense, e.g., Krum and GeoMed, are particularly vulnerable to this attack. Although utilizing the local momentum and gradient decomposition mechanisms can mitigate this type of attack to some extent, further investigations on more effective defenses are still urgently needed.

Training Stability. In our experiments, we found the Clustering method presents the best training stability in both plain distributed SGD and our DeSGD schemes under the most strong ALIE attack, therefore we further plot the learning curves of this method under other typical attacks in Fig. 2. The curves clearly show that the gradient decomposition approach can significantly improve the training stability at the presence of both non-IID data with label skew and Byzantine attacks.

Comparison with Bucketing Mechanism. We also compare our framework with an approach called Bucketing [15] that is proposed for mitigating attacks in the heterogeneous data settings. We choose some popular attacks and defenses, reporting the model accuracy drop compared to the non-Byzantine setting in Table II. From the results we can find that both the Bucketing and gradient decomposition with local momentum can enhance the resilience of various defense methods and each of them has some advantages in specific cases. For

TABLE II: Comparison with the bucketing mechanism. The model accuracy drop over the non-attack setting is reported.

Method	Accuracy Drop (%) (↓)		
	Sign-Flip	IPM	ALIE
M-Krum + Bucketing	2.04	1.53	14.49
M-Krum + DeSGD	1.91	2.05	12.10
TrMean + Bucketing	3.06	1.95	11.59
TrMean + DeSGD	3.80	3.15	10.14
DnC + Bucketing	2.48	1.99	12.10
DnC + DeSGD	2.45	2.16	5.22
Clustering + Bucketing	5.68	2.59	13.27
Clustering + DeSGD	2.06	1.62	1.92

instance, Bucketing could be more suitable for TrMean/DnC defense under Sign-Flip/IPM attack, while DeSGD is generally more robust under the ALIE attack and can improve the performance of Clustering most significantly. Besides, both of them have similar limitations as they implicitly increase the proportion of malicious gradients in each aggregation step.

C. Ablation Studies

In our framework, we apply both local and global special designs for more robust model training. Here, we conduct a set of ablation studies to reveal the importance of each proposed mechanism. First, we will show that using global momentum will increase the vulnerability of model training. From Fig. 3, it can be observed that when global momentum is used, both SGD and DeSGD will result in model divergence. Though local momentum could enhance the robustness, SGD with Krum would also lead to poor performance due to the high bias of estimated gradient under non-IID settings as Krum will only select one local gradient in each iteration. We also find that gradient compensation along with the local momentum could speed up the convergence rate and achieve the best accuracy.

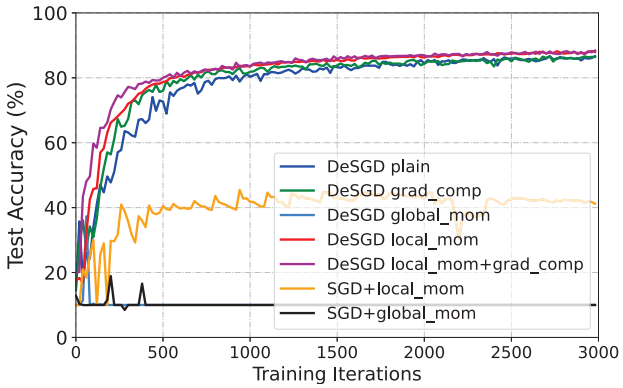


Fig. 3: Test accuracy over iterations with various training strategies under the Sign-Flip attack and Krum defense.

V. Conclusion

In this work, we proposed a novel gradient decomposition based scheme to perform robust distributed learning on non-IID scenarios. We design a learning algorithm with local momentum and global compensation to jointly improve the fidelity and resilience. Numerical results demonstrated the effectiveness of our method against various Byzantine attacks. Future work includes developing jointly communication efficient and robust learning algorithms in decentralized systems.

Acknowledgements

This research is supported in part by the Shenzhen Science and Technology Program under Grant KQTD20170810150821146, National Key R&D Program of China under Grant 2021YFA0715202.

References

[1] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng, "Large scale distributed deep networks," in *NIPS*, 2012.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] Alex Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[4] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, and et al, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[5] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, 2017.

[6] Jeremy Bernstein, Jiawei Zhao, and Kamyar Azizzadenesheli, "signsgd with majority vote is communication efficient and fault tolerant," in *ICLR*, 2019.

[7] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th USENIX Security Symposium*, 2020.

[8] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter L. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *ICML*, 2018.

[9] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault, "The hidden vulnerability of distributed learning in byzantium," in *Proceedings of International Conference on Machine Learning (ICML)*, 2018.

[10] Cong Xie, Sanmi Koyejo, and Indranil Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *ICML*, 2019.

[11] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *Network and Distributed System Security Symposium, NDSS*, 2021.

[12] Gilad Baruch, Moran Baruch, and Yoav Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *NeurIPS*, 2019.

[13] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault, "Distributed momentum for byzantine-resilient stochastic gradient descent," in *International Conference on Learning Representations, ICLR*, 2021.

[14] Avishek Ghosh, Justin Hong, Dong Yin, and Kannan Ramchandran, "Robust federated learning in a heterogeneous environment," *CoRR*, vol. abs/1906.06629, 2019.

[15] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi, "Byzantine-robust learning on heterogeneous datasets via bucketing," in *The Tenth International Conference on Learning Representations, ICLR*, 2022.

[16] Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Rafael Pinot, and John Stephan, "Byzantine machine learning made easy by resilient averaging of momentums," in *ICML*, 2022.

[17] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi, "Learning from history for byzantine robust optimization," in *Proceedings of the 38th International Conference on Machine Learning, ICML*, 2021.

[18] Divyansh Jhunjhunwala, Pranay Sharma, Aushim Nagarkatti, and Gauri Joshi, "Fedvarp: Tackling the variance due to partial client participation in federated learning," in *Uncertainty in Artificial Intelligence*, 2022.

[19] Yudong Chen, Lili Su, and Jiaming Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, 2017.

[20] Qi Xia, Zeyi Tao, Zijiang Hao, and Qun Li, "FABA: an algorithm for fast aggregation against byzantine attacks in distributed neural networks," in *IJCAI*, 2019.

[21] Virat Shejwalkar and Amir Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," *ISOC Network and Distributed Systems Security Symposium*, 2021.

[22] Felix Sattler, Klaus-Robert Müller, Thomas Wiegand, and Wojciech Samek, "On the byzantine robustness of clustered federated learning," in *ICASSP*, 2020.