# Byzantine-resilient Decentralized Stochastic Gradient Descent

Shangwei Guo, Tianwei Zhang, Han Yu, Xiaofei Xie, Lei Ma, Tao Xiang, and Yang Liu

***Abstract*—Decentralized learning has gained great popularity to improve learning efficiency and preserve data privacy. Each computing node makes equal contribution to collaboratively learn a Deep Learning model. The elimination of centralized Parameter Servers (PS) can effectively address many issues such as privacy, performance bottleneck and single-point-failure. However, how to achieve Byzantine Fault Tolerance in decentralized learning systems is rarely explored, although this problem has been extensively studied in centralized systems.**

**In this paper, we present an in-depth study towards the Byzantine resilience of decentralized learning systems with two contributions. First, from the adversarial perspective, we theoretically illustrate that Byzantine attacks are more dangerous and feasible in decentralized learning systems: even one malicious participant can arbitrarily alter the models of other participants by sending carefully crafted updates to its neighbors. Second, from the defense perspective, we propose UBAR, a novel algorithm to enhance decentralized learning with Byzantine Fault Tolerance. Specifically, UBAR provides a Uniform Byzantine-resilient Aggregation Rule for benign nodes to select the useful parameter updates and filter out the malicious ones in each training iteration. It guarantees that each benign node in a decentralized system can train a correct model under very strong Byzantine attacks with an arbitrary number of faulty nodes. We conduct extensive experiments on standard image classification tasks and the results indicate that UBAR can effectively defeat both simple and sophisticated Byzantine attacks with higher performance efficiency than existing solutions.**

***Index Terms*—Decentralized learning, Stochastic gradient descent, Byzantine attack, Byzantine fault tolerance**

## I. INTRODUCTION

THE rapid development of edge computing and Deep Learning (DL) technologies leads to the era of Artificial Intelligence of Things. Nowadays, it is a trend to learn and deploy powerful DL models on edge devices [1]–[5] for various AI tasks (e.g., image classification, video processing). Such collaborative learning can increase the model generalization and achieve data privacy, since the model is trained from different sources of data without being released. Meanwhile, the collaboration mode enables resource-constrained devices to train large-scale models efficiently. One typical example is the federated learning system [6], [7], where multiple edge devices can collaborate to train a shared DL model for different applications and scenarios, such as healthcare [8], security surveillance [9], intelligent transportation [10].

However, federated learning introduces a centralized parameter server, which can bring new security and efficiency drawbacks [7], [11], [12]. First, federated learning suffers from single point of failure. The functionality of the system highly depends on the operations of the parameter server. If the server gets crashed or hacked, then the entire system will be broken down, affecting all the edge devices. Second, the centralized parameter server can be the performance bottleneck, particularly when a large amount of edge devices are connected to this server.

Due to the limitations of PS-based centralized learning, there is a growing trend towards training a DL model in a decentralized fashion [11], [13]–[17]. Specifically, centralized servers are eliminated from the system while each participant plays an equal role (both training and aggregating parameters) in learning the model [18], [19]. This decentralization mode exhibits huge potential for DL applications in many scenarios: in autonomous driving [20], [21], cars can capture images or videos during the driving and collaboratively learn powerful models for detecting traffic lights, sign, lane and pedestrians; in video coding, users can learn faster and better video coding mechanism by communicating with others [22], [23].

A distributed system can be threatened by the famous Byzantine Generals Problem [24]: some nodes inside the network can conduct inappropriate behaviors, and propagate wrong information, leading to the failure of the entire system. This is particularly dangerous in the distributed learning scenarios due to three reasons. (1) Distributed learning requires the collaboration of thousands of edge devices from different domains and parties. It is impossible to guarantee that each device is trusted and reliable. A single dishonest node can send wrong parameters/estimates to affect the entire network and final results. (2) Modern IoT devices and networks are complicated and vulnerable. Recent years have witnessed many infamous IoT attacks (e.g., Mirai Botnet [25], Stuxnet [26]), enabling an adversary to easily compromise a large scale of IoT devices. This facilitates the Byzantine attacks in distributed learning systems. (3) The consequences of attacks can be very severe. Past works have shown that an adversary can compromise the centralized distributed learning system to alter the behaviors of the training process or final models [27].

This Byzantine Generals Problem has been extensively studied in the centralized PS-based learning systems. Attacks with different threat models and goals [28], [29] were designed to demonstrate this vulnerability. Meanwhile, Byzantine-resilient defense solutions were also introduced to enhance the system.

T. Zhang is the corresponding author.

S. Guo and T. Xiang are with College of Computer Science, Chongqing University, Chongqing 400044, China (email: {swguo, txiang}@cqu.edu.cn).

T. Zhang, H. Yu, X. Xie, and Y. Liu are with School of Computer Science and Engineering, Nanyang Technological University 639798, Singapore (email: {tianwei.zhang, han.yu, xfxie, and yangliu}@ntu.edu.sg).

L. Ma is with University of Alberta, Edmonton, Alberta T6G 2R3, Canada (email: ma.lei@acm.org).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3116976, IEEE Transactions on Circuits and Systems for Video Technology



However, very few works have focused on the Byzantine threats in decentralized learning systems. We are particularly interested in two questions: (1) *how feasible and severe are the Byzantine attacks in decentralized learning systems?* (2) *How can we improve the Byzantine resilience of a decentralized system?*. Currently there are no satisfactory answers due to the distinct features of centralized and decentralized systems.

In this paper, we provide an in-depth study to answer the above two questions. First, we formally define the Byzantine Generals Problem in the decentralized learning setting, and theoretically analyze the corresponding vulnerabilities. We discover that the indirect connection to a malicious node cannot reduce the attack cost and amplify the damage. We prove that an adversary can just use one node to alter the models of all nodes inside the system arbitrarily. This is different from the centralized system, which only requires tampering the model on PS for a successful attack.

Second, we explore the possible solutions to secure decentralized learning systems with Byzantine Fault Tolerance (BFT). It is challenging to apply the Byzantine-resilience methods from PS-based systems [27], [30]–[35] to the decentralized scenario due to two reasons. First, those defenses have security and efficiency drawbacks in protecting centralized systems. They are either vulnerable to elaborately designed Byzantine attacks [29], [36], [37], or have large computation overhead and scalability issue with unrealistic requirements (e.g., the PS has extra validation dataset) [28], [29]. These limitations still exist if the defenses were extended into decentralized systems. The second reason lies in the huge differences between centralized and decentralized learning systems. Existing defenses are mainly designed for the centralized PS to make decisions. However, each participant in decentralized learning acts as not only a worker node, but also a PS. In addition, the number of neighbors connected to each node varies dramatically. So some assumptions made in the centralized defenses will not hold. To the best of our knowledge, currently there are few research papers [38] attempting to achieve Byzantine-resilient decentralized learning by comparing the distances among estimates, which is vulnerable to sophisticated Byzantine attacks, as demonstrated in the evaluation section of this paper.

We propose UBAR, a novel **U**niform **B**yzantine-resilient **A**ggregation **R**ule to secure decentralized learning systems. UBAR consists of two design stages. The first stage is introduced to mitigate simple Byzantine attacks (e.g., [30]) by shortlisting a set of candidate nodes: each benign node selects a number of potential benign nodes based on the distances of their parameters to its own. The second stage is used to select the final parameters and defeat advanced Byzantine attacks [29], [36], [37]: each benign node uses its training samples to test the performance of the parameters from the first phase, and chooses the ones with the best training quality.

UBAR leverages the unique features of decentralized systems to overcome the limitations of prior solutions. Since each node acts as both a worker and PS, it can use its own parameters as the baseline (Stage 1) instead of the average or median of neighbor nodes in PS-based systems. This can effectively protect the baseline values from being manipulated,

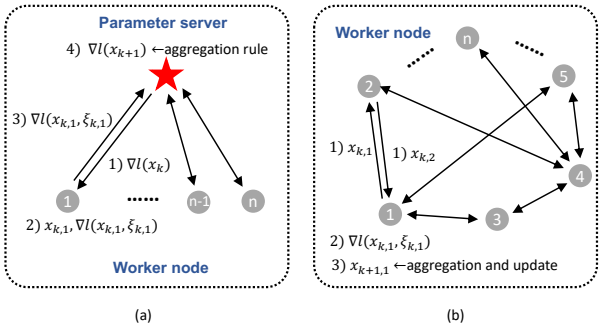



Fig. 1: Distributed learning in centralized (a) and decentralized (b) fashions.

and mitigate an arbitrary number of Byzantine nodes. Each node also uses its training samples for performance evaluation (Stage 2), which can perfectly relax the unrealistic assumption of the server's availability of validation datasets in PS-based systems. Besides, since Stage 1 is vulnerable to advanced Byzantine attacks [29], [36], [37] while Stage 2 suffers from scalability and cost issues, the integration of these two stages can achieve both efficiency and strong Byzantine resilience. We conduct comprehensive experiments to show that UBAR is tolerant against both simple and sophisticated attacks, while all existing defense solutions fail. Besides, UBAR also achieves 8-30X performance improvement over existing methods.

The key contributions of this paper are:

- We theoretically analyze and demonstrate the vulnerabilities of the Byzantine Generals Problem in decentralized learning.
- We propose UBAR, a uniform Byzantine-resilient aggregation rule, to defeat an arbitrary number of Byzantine nodes in decentralized systems.
- We conduct extensive experiments to show UBAR outperforms other solutions for both security and performance.

The rest of this paper is organized as follows. Background and related works are reviewed in Section II. Section III gives formal definitions of decentralized systems We analyze the Byzantine Fault of decentralized learning in Section IV. Section V presents our novel Byzantine-resilient solution. Section VI shows the experimental results under various attacks and system settings. Section VII and VIII discusses the limitations and concludes the paper.

## II. Background and Related Work

### A. Byzantine-resilient Centralized Learning

A centralized learning system consists of a Parameter Server (PS) and multiple distributed worker nodes, as shown in Figure 1(a). Every worker node has its own training dataset, but adopts the same training algorithm. In each iteration, a worker node 1) pulls the gradient from the PS, 2) updates the gradient based on its local data, 3) uploads the new gradient to the PS, and the PS 4) aggregates all the received gradients from the worker nodes into one gradient vector. The nodes repeat the above steps from the new gradient, until the training process is terminated and a model is produced.

Dishonest nodes can compromise the training process and the final model by uploading wrong gradients [29], [36], [37]. It is necessary for the PS to detect such Byzantine nodes and discard their updates when aggregating the gradients.

Motivated by the parameter difference between benign and malicious estimates, a number of solutions cluster the uploaded gradients and detect the outliers based on the vector distances. For instance, Blanchard et al. proposed Krum [27], which chooses the gradient vector with the minimal sum of squared distances to its neighbors as the aggregated one. Median-based Aggregation rules [30], [39] were designed, which inspect the gradient vectors and calculate the median values in each dimension to defeat Byzantine attacks. Mhamdi et al. introduced Bulyan [36], to further enhance existing Byzantine-resilient aggregation rules by combining Krum and Median-based aggregation rules. Although these defenses can defeat simple Byzantine attacks such as Gaussian and bit-flip attacks [27], [30], they were vulnerable against more sophisticated attacks [29], [37]. The reason lies in the vulnerability of distance-based strategies: the close distance between two gradients does not imply similar performance. Thus, these sophisticated attacks could create gradients that are malicious but indistinguishable from benign gradients in distance.

Some solutions selects the benign nodes by evaluating the performance of each uploaded gradient on extra validation datasets. For instance, Xie et al. proposed Zeno [28] and Zeno++ [40] for synchronous and asynchronous learning systems, respectively. Both Zeno and Zeno++ calculate the prediction accuracy of each gradient on the extra validation datasets to identify Byzantine nodes. However, they require that the PS has a validation dataset, which is not realistic under some circumstances. Besides, performance evaluations of all gradients have much more overhead than parameter evaluations. This can significantly increase the total training time and the computation burdens for the PS, especially when the number of worker nodes is larger.

Some solutions select benign gradients and nodes based on their history records. For instance, Hidden Markov Model was utilized [41] to learn the quality of parameter updates during distributed training. The learned profiles can improve the efficiency and accuracy of detecting malicious nodes. Pan et al. [42] utilized the historical interactions with the workers as experience to identify Byzantine attacks via reinforcement learning techniques. However, these solutions cannot guarantee Byzantine resilience. An adversary can easily bypass the detection algorithms by pretending to be benign at the beginning and only uploading malicious parameters at the last several iterations. Then the learned profiles cannot predict malicious behaviors in future iterations.

### B. Byzantine-resilient Decentralized Learning

Decentralized learning systems remove the PS, as every node in the network is also responsible for model update [11], [19], [43]. The architecture of a decentralized learning system is illustrated in Figure 1(b). Specifically, in each iteration of the training process, each worker node 1) broadcasts its parameter vectors (estimates[1]) to its neighbor nodes, and receive the estimates from them; 2) trains the model estimates using the local data. 3) It then aggregates them with the neighbor nodes' estimates and updates the model.

Compared to centralized learning, research of Byzantine-resilient decentralized learning is still at an early stage. Several attempts have been made to achieve Byzantine-resilient decentralized learning [38], [44], [45]. For example, Yang et al. proposed ByRDiE [38] and BRIDGE [44], which simply apply the trimmed-median algorithm from centralized systems [30], [39] to decentralized systems. While ByRDiE is designed for the coordinate descent optimization algorithm, BRIDGE is used in decentralized learning systems with SGD. Similar to [30], [39], those solutions are vulnerable to some Byzantine attacks [36], [37]. In Section VI, we will demonstrate their incapability of defeating sophisticated Byzantine attacks.

Yang and Bajwa [13] proposed RD-SVM to support distributed Support Vector Machine (SVM) against Byzantine attacks. RD-SVM compares the losses from neighbor nodes to identify and filter potential Byzantine nodes. It adopts the hinge loss, and involves all the data samples at each iteration for Byzantine identification. Hence, it is more applicable to linear classifiers like SVM. In contrast, this paper focuses on deep learning models with the mainstream SGD algorithm and batch training feature. We propose UBAR to fulfill these requirements and improving the efficiency.

## III. System Model of Decentralized Learning

In this section, we formally define a decentralized communication system and describe the learning task.

### A. Decentralized Systems

A decentralized system is defined as an undirected graph: $\mathcal{G} = (V, E)$, where $V$ denotes a set of $n$ nodes and $E$ denotes a set of edges representing communication links. Specifically, we have

- $(i, j) \in E$ if and only if node $i$ can receive information from node $j$;
- $(j, i) \in E$ if $(i, j) \in E$.

Let $\mathcal{N}_i = \{j | (i, j) \in E\}$ be the set of the neighbors of node $i$. We further assume that $n_b$ out of $n$ nodes are benign and the rest are malicious. We can define a subgraph that only contains the benign nodes:

**Definition 1.** *(Benign Induced Subgraph) The benign induced subgraph, $\mathcal{G}_b = (V_b, E_b)$, is a subgraph of $\mathcal{G}$, formed by all the benign nodes in $\mathcal{G}$ and all the edges connecting those benign nodes. Specifically,*

- *$i \in V_b \subseteq V$ if $i$ is a benign node and $|V_b| = n_b$;*
- *$(i, j) \in E_b \subseteq E$ if and only if $i, j \in V_b$;*
- *$(j, i) \in E_b$ if $(i, j) \in E_b$.*

Following the information exchange models in [18], [19], we assume the benign induced subgraph is fully connected, i.e., giving two arbitrary benign nodes $i$ and $j$, there always

[1]"Parameter vector" and "estimate" are used interchangeably.

exists at least one path that connects these two nodes. We formally state the assumption as below:

**Assumption 1.** *(Connectivity of Benign Induced Subgraph) There exists an integer $\tau$ such that for $\forall i, j \in V_b$, node $j$ can propagate its information to node $i$ through at most $\tau$ edges.*

### B. Model Training

In a decentralized learning system, $n$ nodes cooperatively train a model by optimizing the loss function with SGD and exchanging estimates with their neighbors. Let $x \in \mathbb{R}^d$ be the $d$-dimensional estimate vector of a DL model; $l$ be the loss function. Each node $i \in V$ obtains a training dataset $D_i$, consisting of independent and identically distributed (IID) data samples from a distribution $D$. Those $n$ nodes train a shared model by solving the following optimization problem.

$$\min_{x \in \mathbb{R}^d} \mathbb{E}_{\xi \sim D} l(x, \xi) \tag{1}$$

where $\xi$ is a training data sample from $D$ and $l(x, \xi)$ is calculated on $\xi$.

## IV. Byzantine Attack in Decentralized Learning

In this section, we theoretically demonstrate the feasibility and severity of Byzantine attacks in decentralized learning systems. Following the decentralized network topology, the nodes in $V$ iteratively optimize the shared model until reaching convergence or the maximum number of iterations. Specifically, at the $k$-th iteration, node $i$ has its local estimate denoted as $x_{k,i}$, and broadcasts it to its neighbors. When receiving the estimates from the neighbors, node $i$ will update its local estimate according to the General Update Function (GUF):

**Definition 2.** *(GUF) Let $x_{k,i}$, $\nabla l(x_{k,i}, \xi_{k,i})$ be the estimate and gradient of node $i$ at the $k$-th iteration. $\{x_{k,j}, j \in \mathcal{N}_i\}$ are the estimates from its neighbors. $\mathcal{R}$ is an aggregation rule. Node $i$ updates its estimate for the $(k+1)$-th iteration using the following general update function:*

$$x_{k+1,i} = \alpha x_{k,i} + (1-\alpha)\mathcal{R}(x_{k,j}, j \in \mathcal{N}_i) - \lambda \nabla l(x_{k,i}, \xi_{k,i}) \tag{2}$$

*where $\lambda$ is the learning rate; $\alpha$ is a hyper-parameter that balances the weights of the estimates.*

Without loss of generality, we assume all the nodes have the same learning rate. The stochastic gradient can be replaced with a mini-batch of stochastic gradients [11].

A straightforward and common way is the average aggregation rule:

$$\mathcal{R}_{Average} = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} x_{k,j} \tag{3}$$

and $\alpha$ is set as $\frac{1}{|\mathcal{N}_i|+1}$. However, because the average aggregation in Equation 3 does not consider BFT, this training process can be easily compromised by Byzantine attacks: an adversary can use just one malicious node to send wrong estimates to its neighbors and alter their aggregated estimates. More seriously, due to the fully connectivity of benign induced subgraph (Assumption 1), this fault will be also propagated to other benign nodes not directly connected to this Byzantine node after several iterations, and finally all the nodes in this network will be affected.

**Theorem 1.** *Consider a decentralized system under the average aggregation rule $\mathcal{R}_{Average}$. In this system $\hat{i}$ is a Byzantine node, attempting to add a malicious vector $\hat{x}$ to the estimate of a benign node $i_{\tau'}$. The shortest distance (i.e., number of edges) between them is $\tau'$ and $\{i_s\}_{s=1}^{\tau'-1}$ are the benign nodes on the shortest trace between $i_{\tau'}$ and $\hat{i}$. The distance between node $i_s$ and $\hat{i}$ is $s$. Then at the $k_0$-th iteration, node $\hat{i}$ can broadcast to its neighbors the following estimate to achieve this goal in $\tau'$ iterations:*

$$x = x_{k_0,\hat{i}} + \hat{x} \prod_{s=1}^{\tau'} (|\mathcal{N}_{i_s}| + 1) \tag{4}$$

*where $|\mathcal{N}_{i_s}|$ is the number of neighbors of node $i_s$.*

*Proof.* We assume that there is only one path of $\tau'$ edges from node $\hat{i}$ to $i_{\tau'}$. We prove the theorem by mathematical induction.

If $\tau' = 1$, the two nodes are neighbors. Then, the estimate of node $i_1$ at $k_0 + 1$ iteration is

$$\hat{x}_{k_0+1,i_1} = \frac{1}{|\mathcal{N}_{i_1}| + 1}(x_{k_0,i_1} + \sum_{j \in \mathcal{N}_{i_1}/\hat{i}} x_{k_0,j} + x_{k_0,\hat{i}} + \hat{x}(|\mathcal{N}_{i_1}| + 1)) - \lambda \nabla l(x_{k_0,i_1}, \xi_{k_0,i_1}) \tag{5}$$

$$= \frac{1}{|\mathcal{N}_{i_1}| + 1}(x_{k_0,i_1} + \sum_{j \in \mathcal{N}_{i_1}} x_{k_0,j}) - \lambda \nabla l(x_{k_0,i_1}, \xi_{k_0,i_1}) + \hat{x} \tag{6}$$

It proves that the theorem is true when $\tau' = 1$.

We now assume the theorem is true when $\tau' = k$, i.e. node $\hat{i}$ sends

$$\hat{x}_{k_0,\hat{i}} = x_{k_0,\hat{i}} + \hat{x} \prod_{s=1}^{k} (|\mathcal{N}_{i_s}| + 1). \tag{7}$$

to its neighbors at $k_0$-th iteration. Then, at $(k_0+k)$-th iteration, node $i_k$'s estimate is

$$\hat{x}_{k_0+k,i_k} = x_{k_0+k,i_k} + \hat{x} \tag{8}$$

where $x_{k_0+k,i_k}$ is the benign estimate that node $\hat{i}$ should send to its neighbors when it was not controlled.

Consider the $k+1$ case. Node $\hat{i}$ sends

$$\hat{x}_{k_0,\hat{i}} = x_{k_0,\hat{i}} + \hat{x} \prod_{s=1}^{k+1} (|\mathcal{N}_{i_s}| + 1) \tag{9}$$

to its neighbors at the $k_0$-th iteration. At the $(k_0 + k)$-th iteration, the estimate of node $i_k$ is

$$\hat{x}_{k_0+k,i_k} = x_{k_0+k,i_k} + \hat{x}(|\mathcal{N}_{i_{k+1}}| + 1). \tag{10}$$

Then, at the $(k_0 + k + 1)$-th iteration, the estimate of node

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3116976, IEEE Transactions on Circuits and Systems for Video Technology



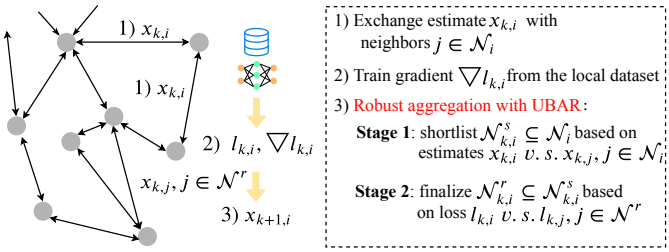



Fig. 2: A decentralized learning system with UBAR.

$x_{k+1}$ is affected:

$$\begin{aligned}
&\hat{x}_{k_0+k+1,i_{k+1}} \\
&= \frac{1}{|\mathcal{N}_{i_{k+1}}|+1}(x_{k_0+k,i_{k+1}} + \sum_{j\in\mathcal{N}_{i_{k+1}}/i_k} x_{k_0+k,j} \\
&+ x_{k_0+k,i_k} + \hat{x}(|\mathcal{N}_{i_{k+1}}|+1)) - \lambda\nabla l(x_{k_0+k,i_{k+1}}, \xi_{k_0+k,i_{k+1}}) \\
&= \frac{1}{|\mathcal{N}_{i_{k+1}}|+1}(x_{k_0+k,i_{k+1}} + \sum_{j\in\mathcal{N}_{i_{k+1}}} x_{k_0+k,j}) \\
&- \lambda\nabla l(x_{k_0+k,i_{k+1}}, \xi_{k_0+k,i_{k+1}}) + \hat{x}
\end{aligned} \tag{11}$$

□

## V. Byzantine-resilient Solution

### A. Byzantine-resilient Aggregation Rule.

Due to the Byzantine threat of decentralized learning, it is necessary to design a robust aggregation rule to defeat Byzantine nodes. This rule should guarantee that all benign nodes converge to the optimal estimate learned without Byzantine nodes. In the following, we propose UBAR, a novel aggregation rule for decentralized systems to satisfy the above requirement and uniformly defend against Byzantine attacks.

### B. UBAR

The design of UBAR is motivated by three observations. First, as introduced in Section II, existing Byzantine defenses for centralized systems have certain security vulnerabilities or practical limitations. Such design flaws still exist when we extend the solutions to decentralized scenarios. Second, a decentralized system has higher convergence requirement than a centralized system: convergence of one parameter server enforced by the solutions cannot guarantee the convergence of all benign nodes in a decentralized system. Third, centralized Byzantine-resilient solutions usually assume a fixed number of faulty nodes connected to the parameter server, while in a decentralized system, the number of faulty nodes connected to each benign node varies significantly. As such, it is necessary to have a more robust Byzantine-resilient solution that can defeat *an arbitrary number* of Byzantine nodes and guarantee convergence of *each benign node* in decentralized systems.

UBAR aims to achieve this goal and overcome the above limitations. It consists of two stages for each training iteration as shown in Fig. 2. At the first stage, each benign node selects a candidate pool of potential benign nodes from its neighbors. The selection is made by comparing the Euclidean distance of the estimate of each neighbor node with its own estimate. One innovation of this stage is the benign node uses its own parameter as the baseline value instead of the median or mean value of its neighbors' parameters as in centralized PS-based systems [27], [30], [39]. This is based on the unique feature of decentralized systems that each node is responsible for both training and aggregation. It gives stronger Byzantine resilience as the baseline values trained from local datasets can never be manipulated by Byzantine nodes, while the aggregated parameter can be poisoned according to Theorem 1 Although after this stage, the candidate pool might still contain Byzantine nodes as the distance-based strategies are not strict Byzantine-resilient, it indeed reduces the scope of benign nodes for further selection.

At the second stage, each benign node further picks the final nodes from the candidate pool for estimate update. It reuses the training sample as the validation set to test the performance (i.e., loss function value) of each estimate. It selects the estimates whose loss values are smaller than its own estimate, and calculates the average of those estimates as the final updated value. One novelty of this stage is the adoption of training samples for performance evaluation of neighbors' parameters. In contrast, prior works in centralized systems require the PS to have an extra validation dataset for evaluation, which may not be applicable in certain scenarios.

It is interesting to note that the selection criteria at Stage 1 is still vulnerable to advanced attacks [29], [36], [37], while the strategy at Stage 2 has efficiency and scalability issues especially when the connectivity is high. By integrating them into one approach, Stage 2 can help Stage 1 further defeat the advanced attacks, while Stage 1 can reduce the computation cost at Stage 2, as it decreases the size of candidate nodes for evaluation. UBAR can be formally described as below:

**Definition 3.** *(UBAR) Let $x_{k,i}$ be the estimate of node $i$ at the $k$-th iteration; $l_{k,i}$ be the loss of the estimates on the stochastically selected data sample, i.e., $l_{k,i} = l(x_{k,i}, \xi_{k,i})$; $\rho_i$ be the ratio of benign neighbors of node $i$. The proposed **U**niform **B**yzantine-resilient **A**ggregation **R**ule,* UBAR*, is define as*

$$\mathcal{R}_{\text{UBAR}} = \begin{cases} \frac{1}{|\mathcal{N}^r_{k,i}|}\sum_{j\in\mathcal{N}^r_{k,i}} x_{k,j}, & \text{if } \mathcal{N}^r_{k,i} \neq \emptyset \\ x_{k,j^*}, & \text{Otherwise} \end{cases} \tag{12}$$

*where*

$$\text{(Stage 1) } \mathcal{N}^s_{k,i} = \underset{\substack{\mathcal{N}^*\subseteq\mathcal{N}_i \\ |\mathcal{N}^*|=\rho_i|\mathcal{N}_i|}}{\operatorname{argmin}} \sum_{j\in\mathcal{N}^*} ||x_{k,j} - x_{k,i}||,$$

$$\text{(Stage 2) } \mathcal{N}^r_{k,i} = \bigcup_{\substack{j\in\mathcal{N}^s_{k,i} \\ l_{k,j}\leq l_{k,i}}} j, \text{ and } j^* = \underset{j\in\mathcal{N}^s_{k,i}}{\operatorname{argmin}} l_{k,j}.$$

Algorithm 1 details the training process of node $i$ using UBAR in a decentralized system. The algorithm begins with the estimate $x_{0,i} = x_0$. At the $k$-th iteration, node $i$ broadcasts its estimate to and receives the estimates from its neighbors. It stochastically selects a training data sample $\xi_{k,i}$ and calculates the loss and the gradient (Lines 3-4). Then it conducts two-stage estimate selection. First, it calculates the Euclidean distances between $x_{k,i}$ and the estimates from its neighbors

**Algorithm 1:** The training algorithm for each benign node $i$ using UBAR.

**Input:** Initial estimate $x_0$, learning rate $\lambda$, number of iterations $K$, ratio of benign nodes $\rho_i$

1 **for** $k$ *in* $[0, K)$ **do**
2 Broadcast $x_{k,i}$ and receive $x_{k,j}$ from $j \in \mathcal{N}_i$;
3 Stochastically sample $\xi_{k,i}$ from $D_i$ ;
4 $l_{k,i} \leftarrow l(x_{k,i}, \xi_{k,i})$ and compute the local gradient $\nabla l_{k,i}$ ;
5 **for** $j$ *in* $\mathcal{N}_i$ **do**
6 $d_{i,j} \leftarrow ||x_{k,i} - x_{k,j}||$;
7 $\mathcal{N}^s_{k,i} \leftarrow \operatorname{argmin}_{\mathcal{N}^* \subseteq \mathcal{N}_i, |\mathcal{N}^*| = \rho_i |\mathcal{N}_i|} \sum_{j \in \mathcal{N}^*} d_{i,j}$ ;
8 **for** $j \in \mathcal{N}^s_{k,i}$ **do**
9 $l_{k,j} \leftarrow l_{k,i}$;
10 **if** $l_{k,i} - l_{k,j} \geq 0$ **then**
11 append $j$ to $\mathcal{N}^r_{k,i}$ ;
12 **if** $\mathcal{N}^r_{k,i}$ *is* $\emptyset$ **then**
13 $j^* \leftarrow \operatorname{argmin}_{j \in \mathcal{N}^s_{k,i}} l_{k,j}$;
14 append $j^*$ to $\mathcal{N}^r_{k,i}$ ;
15 $\mathcal{R}_{k,i} \leftarrow \frac{1}{|\mathcal{N}^r_{k,i}|} \sum_{j \in \mathcal{N}^r_{k,i}} x_{k,j}$ ;
16 Update the local estimate $x_{k+1,i} \leftarrow \alpha x_{k,i} + (1-\alpha)\mathcal{R}_{k,i} - \lambda \nabla l_{k,i}$ ;
17 **return** $x_{K,i}$

and selects $\rho_i|\mathcal{N}_i|$ neighbors with lowest distances (Lines 5-7).

Second, for each estimate $x_{k,j}, j \in \mathcal{N}^s_{k,i}$, node $i$ calculates the loss of $x_{k,j}$ on $\xi_{k,i}$. It chooses the estimates that have similar or better performance than that of $x_{k,i}$ (Lines 8-14). Finally it calculates the average value of the selected nodes and updates the final estimate using GUF (Lines 15-16).

### C. Complexity Analysis

The training process with UBAR is performance efficient, as proved below:

**Proposition 1.** *(Cost of* UBAR*) The computational complexity of* UBAR *is $O(|\mathcal{N}_i|d)$ for each node at each iteration, where $d$ is the dimension of the estimate vector.*

*Proof.* For node $i \in V_b$, at each iteration, UBAR aggregates the received estimates with three operations. First, UBAR selects $\rho_i|\mathcal{N}_i|$ neighbors that are closest to its current estimate. The cost is $O(|\mathcal{N}_i|d)$. Second, UBAR calculates the loss of the selected estimates on the stochastic sample and the cost is $O(\rho_i|\mathcal{N}_i|d)$. Finally, UBAR takes at most $O(\rho_i|\mathcal{N}_i|d)$ to aggregate the estimates with better performance. Since $\rho_i \leq 1$ for $\forall i \in V_b$, the overall computational complexity of UBAR is $O(|\mathcal{N}_i|d)$. □

Compared to existing aggregation rules, the complexity of Average aggregation, Median-based and BRIDGE is $O(|\mathcal{N}_i|d)$, while Krum and Bulyan have a complexity of $O(|\mathcal{N}_i|^2 d)$. So we conclude that UBAR maintains the same performance efficiency as some solutions and performs much better than others, especially when the number of connected neighbor nodes becomes large.

## VI. EXPERIMENTS

### A. Experimental Setup and Configurations

**Datasets.** We evaluate our defense solution with a DL-based image classification task. Specifically, we train a Convolutional Neural Network (CNN) over the MNIST and CIFAR10 datasets [46]. This CNN includes two max–pooling layers and three fully connected layers [36]. We adopt a batch size of 256 and a fading learning rate $\lambda(k) = \lambda_0 \frac{20}{20+k}$ where $k$ is the number of epochs and the initial learning rate $\lambda_0 = 0.05$.

**Implementation of decentralized systems.** The network topology of the decentralized system in our consideration is defined by a connection rate between the nodes. A connection rate is the probability that a node is connected to another node. To ensure the connectivity assumption, we first generate a decentralized network in which all the nodes strictly follow the learning procedure. Then we randomly add Byzantine nodes to the network. To simulate various adversarial environments, we adopt a new parameter, Byzantine ratio, which is defined as the number of Byzantine nodes divided by the number of all nodes in the network. We assume that the Byzantine ratio of node $i$ is lower than $1 - \rho_i$. Without lose of generality, we set $\rho_i$ as 0.4 for all benign nodes. We train the deep learning model in a synchronous mode. We simulate the operations of the decentralized system by running the nodes serially at each iteration. All our experiments are conducted on a server equipped with Xeon Silver 4214 @2.20 GHz CPUs and a NVIDIA Tesla P40 GPU.

**Baselines.** Since there are very few works focusing on decentralized learning systems, we can extend existing aggregation rules from centralized systems to the decentralized case as our baselines, since the estimates and gradients have the same dimensionality and network structure. We consider three popular Byzantine-resilient solutions: Krum [27], marginal median [30], [39], and Bulyan [36].

It is worth noting that in a centralized distributed system, the maximal number of Byzantine workder nodes connected to the parameter server is usually assumed. This does not hold in a decentralized system, as the number of Byzantine nodes connected to each benign node varies greatly. To meet this assumption, we approximately calculate the number of Byzantine nodes allowed for each benign node in Equation 13. In this equation, $\hat{n}_i$ is the number of Byzantine nodes connected to node $i$, $\rho_{central}$ is the maximal ratio of Byzantine nodes allowed in a centralized distributed defense, and $\lceil \cdot \rceil$ is the ceil function.

$$\hat{n}_i = \lceil |\mathcal{N}_i| \cdot \min\{1 - \rho_i, \rho_{central}\} \rceil \tag{13}$$

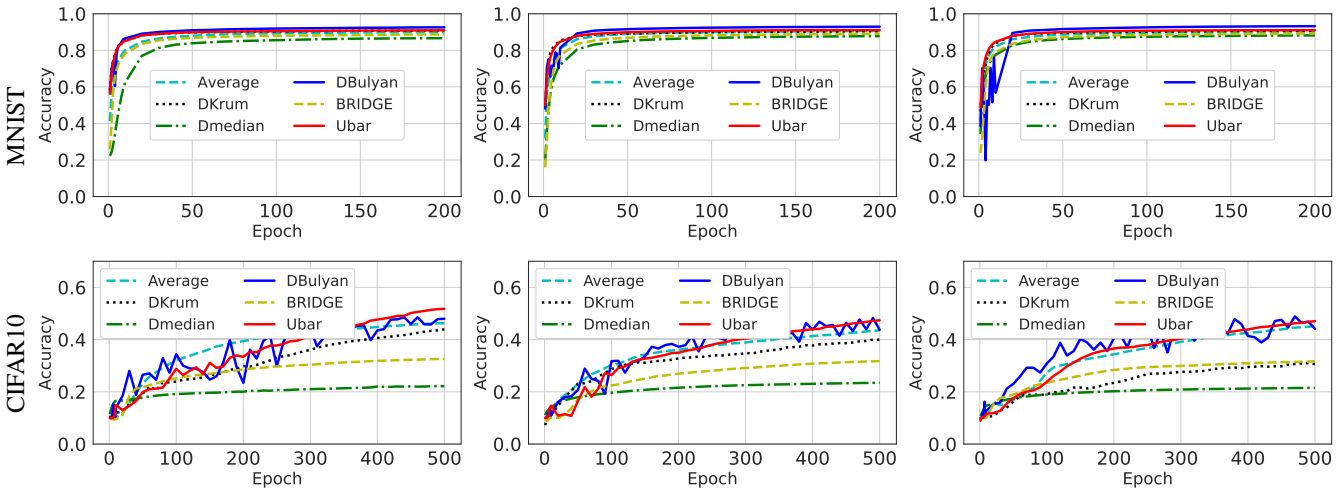
**DKrum.** Similar to Krum, let $j \neq j'$ be two neighbors of node $i$ and we denote $j \to j'$ the $|\mathcal{N}_i| - \hat{n}_i + 2$ closest estimate vectors to the estimate of node $j$. Then, we calculate the score of each neighbor:

$$s(j) = \sum_{j \to j'} ||x_j - x_{j'}||^2$$

We select the estimate with the minimal score as the aggregated estimate. Formally, the aggregated rule DKrum is

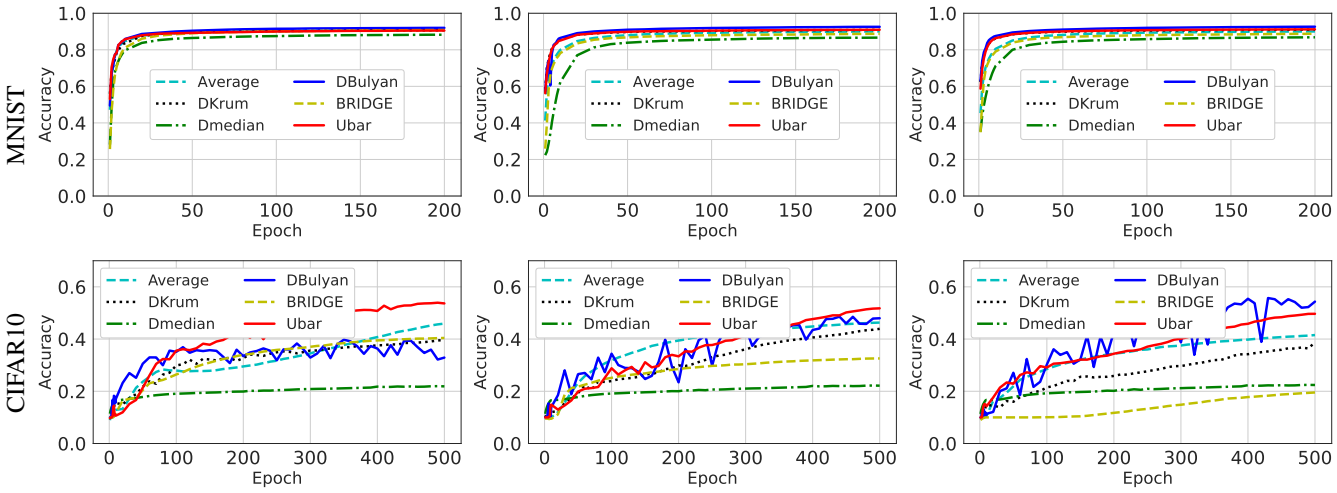
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3116976, IEEE Transactions on Circuits and Systems for Video Technology






(a) 30 nodes (b) 50 nodes (c) 70 nodes

Fig. 3: The worst accuracy of the benign nodes with different network sizes.




(a) Connection ratio = 0.2 (b) Connection ratio = 0.4 (c) Connection ratio = 0.6

Fig. 4: The worst accuracy of the benign nodes with different connection ratios.

defined as

$$\mathcal{R}_{DKrum} = x_{j^*},\ j^* = \underset{j \in \mathcal{N}_i}{\operatorname{argmin}}\, s(j).$$

**Dmedian.** To apply the marginal median solution [30], [39] to decentralized systems, we only need to replace the gradients with the received estimates. Specifically, the aggregated rule $Dmedian$ is defined as

$$\mathcal{R}_{Dmedian} = \texttt{MarMed}\{x_j, j \in \mathcal{N}_i\} \tag{14}$$

where `MarMed` is the marginal median function defined in [30], [39]. Informally, the $m$-th dimensional value of $\mathcal{R}_{Dmedian}$ is the median of the $m$-th dimensional elements of all estimates in $\mathcal{N}_i$.

**DBulyan.** At each iteration, node $i$ first recursively uses DKrum to select $|\mathcal{N}_i| - 2\hat{n}_i$ estimates, i.e., $\{x_j, j \in \mathcal{N}_i^{DKrum}\}$, where $|\mathcal{N}_i^{DKrum}| = |\mathcal{N}_i| - 2\hat{n}_i$. Specifically, node $i$ uses DKrum to select one estimate from its neighbors and deletes the corresponding node from its neighbors. Then, node $i$ recursively selects the remaining estimates using DKrum. Finally, it adopts a median-based method to aggregate the estimates in $\{x_j, j \in \mathcal{N}_i^{DKrum}\}$. Formally, the $m$-th coordinate of the aggregated estimate is calculated as

$$\mathcal{R}_{DBulyan}[m] = \frac{1}{\beta} \sum_{j \in \mathcal{M}[m]} x_j[m] \tag{15}$$

where $\beta = |\mathcal{N}_i| - 4\hat{n}_i$ and $\mathcal{M}[m]$ is the set of neighbors with the size of $\beta$. The sum of the $m$-th elements to its median is minimal among all subsets of $\mathcal{N}_i^{DKrum}$ with the size $\beta$.

In addition to the above defenses from centralized systems, we also implement BRIDGE [44] for comparison, which is designed specifically for decentralized systems, i.e., BRIDGE [44]. For all these defenses, we set $\alpha$ to be 0.5 in the GUF. For baseline, we consider the same decentralized system configuration without Byzantine nodes, and using the Average Aggregation rule (Equation 3). The model trained from this setting can be regarded as the optimal one.

**Performance Metric.** For each defense deployed in the decentralized system, we measure the testing accuracy of the trained model on each benign node, and report the worst accuracy among all nodes to represent the effectiveness of this defense.

### *B. Convergence*

As an aggregation rule in a decentralized system, the essential functionality is to achieve uniform convergence, i.e., the model in each benign node must converge to the correct one. We evaluate the convergence functionality of UBAR with different configurations.

**Network size.** We first evaluate the convergence of our solution under different network sizes. It is more difficult to achieve uniform convergence when there are more nodes. In our experiments, we consider a decentralized system with 30, 40 and 50 nodes respectively [36]. The connection ratio is set as 0.4. Fig. 3 shows the worst accuracy during the training phase on MNIST and CIFAR10.

We can observe that only DBulyan and our proposed UBAR have the same convergence as the baseline on both MNIST and CIFAR10. DBulyan and UBAR converge to a slightly better model at a higher speed. In contrast, DKrum and BRIDGE have bad convergence performance on CIFAR10, especially when the network size is larger. Dmedian does not converge on both datasets when the network size is 50. We also observe that Average Aggregation Rule does not perform better than other methods even under the Byzantine-free setting. This is because this baseline needs to consider all the parameters, some of which may have poor performance even they are not malicious. In contrast, other methods selectively aggregate certain parameters with positive contributions to the model convergence, thus exhibiting better robustness.

**Network connection ratio.** This factor can also affect the model convergence: it takes more effort and time for all nodes to reach the consensus when the connection is heavier. We evaluate such impact with different connection ratio (0.2, 0.4 and 0.6), while fixing the number of nodes as 30.

Fig. 4 illustrates that most defense solutions in our consideration have satisfactory convergence performance when the connection ratio is small (0.2 and 0.4). Our proposed UBAR has better convergence performance when the connection ratio is 0.6. The BRIDGE and DMedian approaches cannot produce correct models at this high connectivity.

### *C. Byzantine Fault Tolerance*

We evaluate the performance of different defense strategies under various Byzantine attacks. We set the connection ratio of the evaluated system as 0.4 and the number of benign nodes as 30. We consider different Byzantine ratios (0.1, 0.3 and 0.5).

**Guassian attack.** We first use a simple attack to test the Byzantine resilience. Specifically, in each iteration the adversarial nodes broadcast to their neighbors random estimate vectors following the Gaussian distribution. We refer to this kind of attack as Gaussian attack.

Fig. 5 Illustrates the model training performance under the Gaussian attack. The advantage of UBAR over other strategies is obvious. Dmedian, DBulyan and BRIDGE do not uniformly converge in all systems of CIFAR10. DKrum fails to converge at the Byzantine ratio of 0.5. Only UBAR can generate the correct model regardless of the Byzantine ratio on both datasets.

**Bit-flip attack.** We also implement a bit-flip attack [30] to evaluate these defenses, where at each iteration the adversarial nodes flip the sign of the floating estimates and then broadcast these fault estimates to their neighbors.

Fig. 6 shows the convergence results: the advantage of UBAR over other strategies is more obvious. UBAR has the same performance as the baseline regardless of the Byzantine ratio on both datasets. This indicates that it is absolutely Byzantine-resilient against the bit-flip attack. In contrast, Dmedian (resp. DBulyan) do not converge uniformly when the Byzantine ratio is high (0.3) on CIFAR10 (resp. MNIST). BRIDGE performs unsatisfactory on both datasets and all baselines fail to defeat the bit-flip attack when the Byzantine ratio is 0.5.

**Sophisticated attack.** To fully evaluate the BFT of our proposed approach, we adopt a more sophisticated attack, Mhamdi attack [36]: the adversary has the capability of collecting all the uploaded estimates from other neighbor nodes. Then it can carefully design its own estimate to make it undetectable from the benign ones, while still compromising the training process. Mhamdi attack has been shown effective against most defenses in centralized PS-based systems [36].

The results are shown in Fig. 7. We observe that UBAR can always succeed for different Byzantine ratios on both datasets. In contrast, other solutions fail to defeat Mhamdi attack in some cases: all solutions fail to converge when the Byzantine ratio is 0.5. Dmedian and BRIDGE cannot converge even when the Byzantine ratio is 0.1 on CIFAR10.

### *D. Computation Cost*

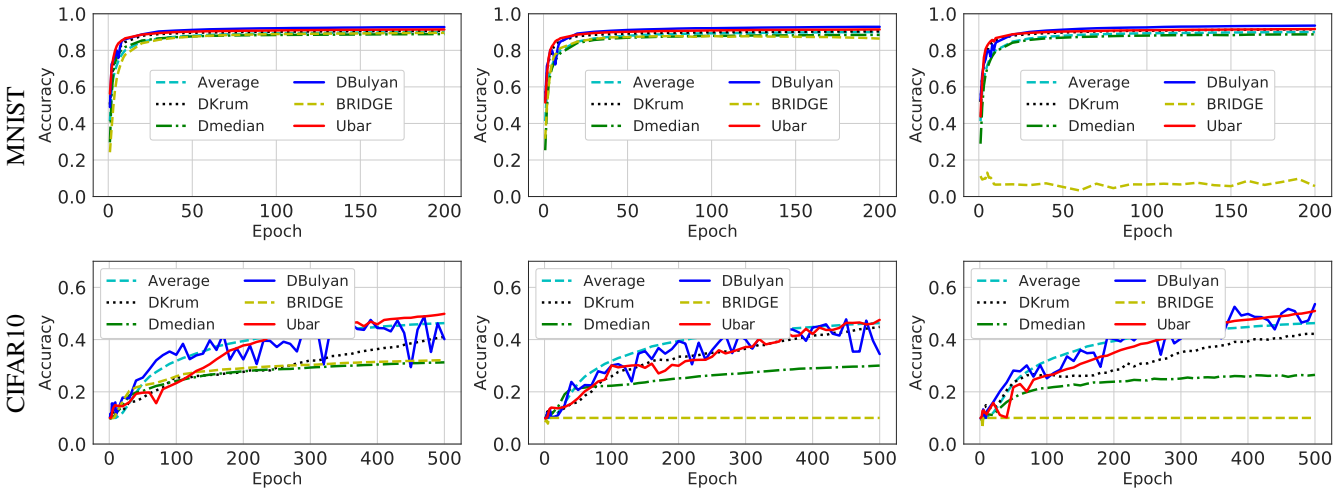
To evaluate the computation cost of UBAR, we measure the average training and aggregation time for one iteration on CIFAR10. We consider a decentralized system with 30 nodes and the connectivity is 0.4. Tab. I shows the average time of different defense solutions. We can see that the training time for those solutions are identical, but the aggregation time differs a lot. UBAR can finish one iteration in a much shorter time than DKrum (8X faster) and DBulyan (30X faster). The reason is that while UBAR only calculates the difference between a node and its neighbors, DKrum and DBulyan have to compare the distances among all neighbors. UBAR is slightly worse than Dmedian and BRIDGE. But considering the bad convergence of Dmedian and BRIDGE under Byzantine attacks demonstrated in Section VI-C, we conclude that UBAR is the optimal solution with the strongest Byzantine resilience and acceptable computation overhead.

## VII. DISCUSSION

**Ratio of Benign Neighbors**. It is worth noting that each benign node needs to know the ratio of benign neighbors to perform robust aggregation. This assumption is commonly adopted in prior works [27], [28], [44]: the nodes can assess

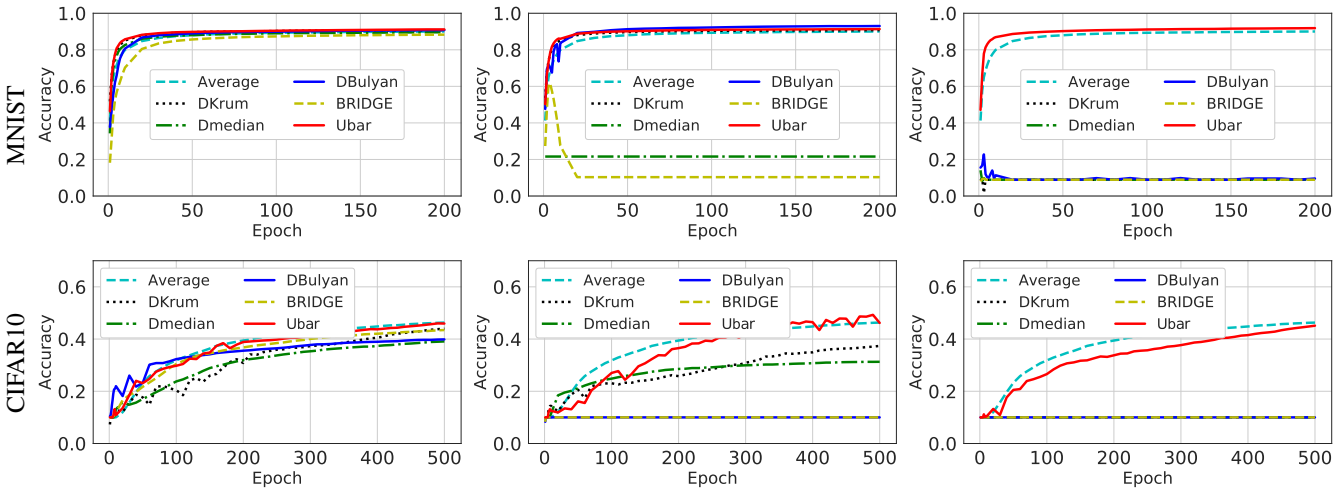
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3116976, IEEE Transactions on Circuits and Systems for Video Technology






(a) Byzantine ratio = 0.1 (b) Byzantine ratio = 0.3 (c) Byzantine ratio = 0.5

Fig. 5: The worst accuracy of the benign nodes under the Gaussian attack.




(a) Byzantine ratio = 0.1 (b) Byzantine ratio = 0.3 (c) Byzantine ratio = 0.5

Fig. 6: The worst accuracy of the benign nodes under the bit-flip attack.

the threat of the surrounding environment before the training process and sets $\rho_i$ accordingly. In case a node does not have such knowledge, it can conservatively set $\rho_i = \frac{1}{|\mathcal{N}_i|}$, where it just assumes one benign neighbor according to Assumption 1. How to accurately estimate $\rho_i$ and aggregate the parameters in an environmental-agnostic way is beyond of the scope of our work.

**Non-IID Scenario**. In this paper, we mainly consider the setting where all the clients use the IID data samples for collaborative training. This is consistent with other Byzantine-resilience works [27], [30], [36], [38]–[42], [44]. In reality, different clients may use non-IID training data to increase the model generalization. This will increase the difficulty of Byzantine defense, since it is hard for a node to distinguish a malicious neighbor from a benign neighbor using different distributions of samples. How to design a Byzantine-resilience method under the Non-IID scenario is a challenging task, and very few works can achieve such protection[2]. This will be an important research direction as future work.

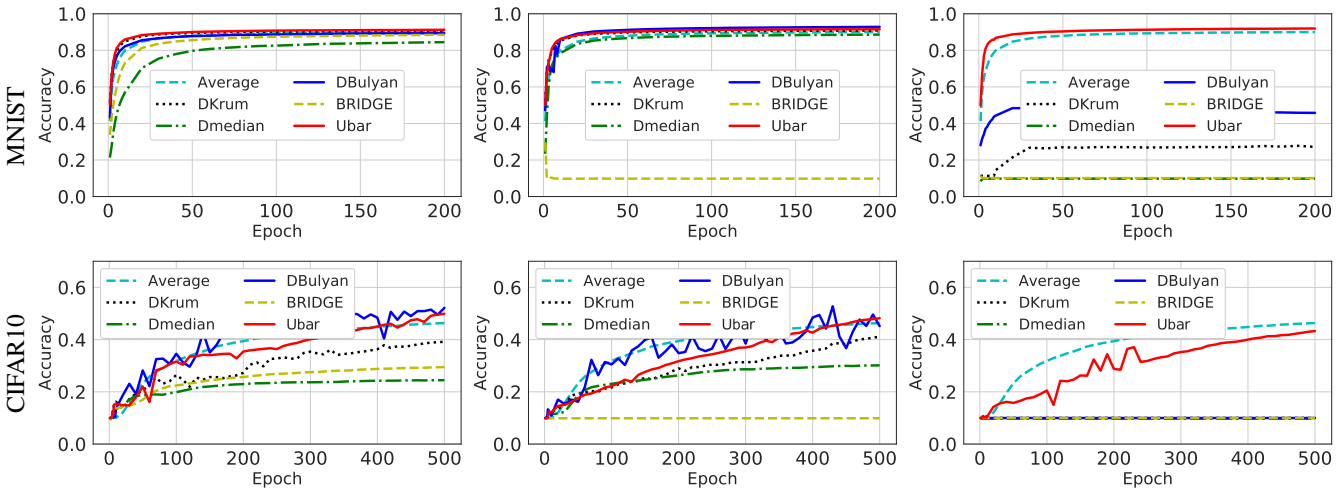
## VIII. Conclusion

In this paper, we explore the Byzantine Fault Tolerance in decentralized learning systems. We demonstrate that a decentralized system is highly vulnerable to Byzantine attacks. We show that existing Byzantine-resilient solutions in centralized PS-based systems cannot be used to protect decentralized systems due to their security flaws and inefficiency. Then we propose a uniform Byzantine-resilient approach, UBAR to defeat Byzantine attacks in decentralized learning. Experimental results reveal that UBAR can resist both simple and sophisticated Byzantine attacks with low computation overhead under different system configurations.

[2]For example, [47] evaluated the Non-IID case. However, this work still adopted the same training set, but just distributed unbalanced samples to different clients.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3116976, IEEE Transactions on Circuits and Systems for Video Technology






(a) Byzantine ratio = 0.1 (b) Byzantine ratio = 0.3 (c) Byzantine ratio = 0.5

Fig. 7: The worst accuracy of the benign nodes under the Mhamdi attack.

TABLE I: The average training and aggregation time of one iteration for different aggregation rules

| Method | Training (s) | Aggregation (s) |
|---|---|---|
| Average | 0.05 | 0.05 |
| Dkrum | 0.05 | 4.36 |
| Dmedian | 0.04 | 0.26 |
| DBulyan | 0.05 | 16.22 |
| BRIDGE | 0.05 | 0.43 |
| UBAR | 0.04 | 0.55 |

## ACKNOWLEDGMENTS

This research was supported in part by the National Natural Science Foundation of China under Grants 62102052; in part by Singapore Ministry of Education Academic Research Fund Tier 1 under Award No. RS02/19 and 2018-T1-002-069; in part by the National Research Foundation, Prime Ministers Office, Singapore under Award No. NRF2018NCR-NCR009-0001, NRF2018NCR-NCR005-0001, NRF2018NCR-NSOE003-0001, NRFI06-2020-0022, and AISG2-RP-2020-019; in part by the Joint NTU-WeBank Research Centre on Fintech under Award No: NWJ-2020-008; in part by the Nanyang Assistant Professorship (NAP); and in part by the RIE 2020 Advanced Manufacturing and Engineering Programmatic Fund, Singapore under Award No. A20G8b0102.

## REFERENCES

[1] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[2] Y. Wang, Z. Jiang, X. Chen, P. Xu, Y. Zhao, Y. Lin, and Z. Wang, "E2-Train: Training state-of-the-art CNNs with over 80% energy savings," in *Advances in Neural Information Processing Systems*, 2019, pp. 5139–5151.

[3] J. Sun, T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen, and Y. Liu, "Stealthy and efficient adversarial attacks against deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[4] J. Chen, K. Li, Q. Deng, K. Li, and S. Y. Philip, "Distributed deep learning model for intelligent video surveillance systems with edge computing," *IEEE Transactions on Industrial Informatics*, 2019.

[5] C. Zhao and A. Basu, "Dynamic deep pixel distribution learning for background subtraction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 11, pp. 4192–4206, 2019.

[6] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

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

[8] O. Rudovic, N. Tobis, S. Kaltwang, B. Schuller, D. Rueckert, J. F. Cohn, and R. W. Picard, "Personalized federated deep learning for pain estimation from face images," *arXiv preprint arXiv:2101.04800*, 2021.

[9] Z. Zhang, S. Wang, Y. Hong, L. Zhou, and Q. Hao, "Distributed dynamic map fusion via federated learning for intelligent networked vehicles," *arXiv preprint arXiv:2103.03786*, 2021.

[10] D. M. Manias and A. Shami, "Making a case for federated learning in the internet of vehicles and intelligent transportation systems," *arXiv preprint arXiv:2102.10142*, 2021.

[11] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 5330–5340.

[12] X. Xie, L. Ma, H. Wang, Y. Li, Y. Liu, and X. Li, "Diffchaser: Detecting disagreements for deep neural networks," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2019, pp. 5772–5778.

[13] Z. Yang and W. U. Bajwa, "RD-SVM: A resilient distributed support vector machine," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 2444–2448.

[14] L. Su and N. H. Vaidya, "Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms," in *ACM Symposium on Principles of Distributed Computing*, 2016, pp. 425–434.

[15] R. Dobbe, D. Fridovich-Keil, and C. Tomlin, "Fully decentralized policies for multi-agent systems: An information theoretic approach," in *Advances in Neural Information Processing Systems*, 2017, pp. 2941–2950.

[16] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Advances in Neural Information Processing Systems*, 2018, pp. 7652–7662.

[17] A. Lalitha, X. Wang, O. Kilinc, Y. Lu, T. Javidi, and F. Koushanfar, "Decentralized bayesian learning over graphs," *arXiv preprint arXiv:1905.10466*, 2019.

[18] J. N. Tsitsiklis, "Problems in decentralized decision making and computation." Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, Tech. Rep., 1984.

[19] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, p. 48, 2009.

[20] T. Chen and S. Lu, "Robust vehicle detection and viewpoint estimation with soft discriminative mixture model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 2, pp. 394–403, 2015.

[21] A. Gulati, G. S. Aujla, R. Chaudhary, N. Kumar, and M. S. Obaidat, "Deep learning-based content centric data dissemination scheme for Internet of Vehicles," in *IEEE International Conference on Communications*, 2018, pp. 1–6.

[22] A. Abou-Elailah, F. Dufaux, J. Farah, M. Cagnazzo, and B. Pesquet-Popescu, "Fusion of global and local motion estimation for distributed video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 1, pp. 158–172, 2012.

[23] J. Yang, L. Qing, W. Zeng, and X. He, "High-order statistical modeling based on a decision tree for distributed video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 5, pp. 1488–1502, 2018.

[24] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[25] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *USENIX Security Symposium*, 2017, pp. 1093–1110.

[26] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

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

[28] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *International Conference on Machine Learning*, 2019, pp. 6893–6901.

[29] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *USENIX Security Symposium*, 2020.

[30] C. Xie, O. Koyejo, and I. Gupta, "Generalized Byzantine-tolerant SGD," *arXiv preprint arXiv:1802.10116*, 2018.

[31] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "DRACO: Byzantine-resilient distributed training via redundant gradients," in *International Conference on Machine Learning*, 2018, pp. 903–912.

[32] C. Xie, S. Koyejo, and I. Gupta, "SLSGD: Secure and efficient distributed on-device machine learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019.

[33] N. Konstantinov and C. Lampert, "Robust learning from untrusted sources," in *International Conference on Machine Learning*, 2019, pp. 3488–3498.

[34] J.-y. Sohn, D.-J. Han, B. Choi, and J. Moon, "Election coding for distributed learning: Protecting SignSGD against byzantine attacks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[35] N. Konstantinov, E. Frantar, D. Alistarh, and C. Lampert, "On the sample complexity of adversarial multi-source pac learning," in *International Conference on Machine Learning*, 2020, pp. 5416–5425.

[36] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *International Conference on Machine Learning*, 2018, pp. 3521–3530.

[37] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 8632–8642.

[38] Z. Yang and W. U. Bajwa, "ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning," *IEEE Transactions on Signal and Information Processing over Networks*, 2019.

[39] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*, 2018, pp. 5650–5659.

[40] C. Xie, O. Koyejo, and I. Gupta, "Zeno++: Robust fully asynchronous SGD," *arXiv preprint arXiv:1903.07020*, 2019.

[41] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," *arXiv preprint arXiv:1909.05125*, 2019.

[42] X. Pan, M. Zhang, D. Wu, Q. Xiao, S. Ji, and M. Yang, "Justinian's gaavernor: Robust distributed learning with gradient aggregation agent," pp. 1641–1658, 2020.

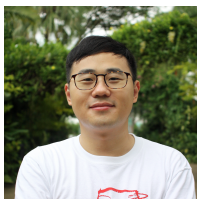
[43] L. He, A. Bian, and M. Jaggi, "COLA: Decentralized linear learning," *Advances In Neural Information Processing Systems*, 2018.

[44] Z. Yang and W. U. Bajwa, "BRIDGE: Byzantine-resilient decentralized gradient descent," *arXiv preprint arXiv:1908.08098*, 2019.

[45] J. Peng and Q. Ling, "Byzantine-robust decentralized stochastic optimization," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 5935–5939.

[46] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[47] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *The Network Distributed System Security Symposium*, 2021.

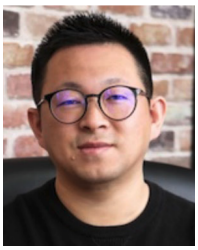



**Shangwei Guo** is an associate professor in College of Computer Science, Chongqing University. He received the Ph.D. degree in computer science from Chongqing University, Chongqing, China at 2017. He worked as a postdoctoral research fellow at Hong Kong Baptist University and Nanyang Technological University from 2018 to 2020. His research interests include secure deep learning, secure cloud/edge computing, and database security.

**Tianwei Zhang** is an assistant professor in School of Computer Science and Engineering, at Nanyang Technological University. His research focuses on computer system security. He is particularly interested in security threats and defenses in machine learning systems, autonomous systems, computer architecture and distributed systems. He received his Bachelor's degree at Peking University in 2011, and the Ph.D degree in at Princeton University in 2017.

**Han Yu** received the BEng (hons) degree and PhD degree from the School of Computer Science and Engineering (SCSE), Nanyang Technological University (NTU), Singapore, in 2007 and 2014, respectively. He is currently a Nanyang assistant professor (NAP) at SCSE, Nanyang Technological University, Singapore. From 2015 to 2018, he held the prestigious Lee Kuan Yew post-doctoral Fellowship (LKY PDF) at the Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY). His research focuses on the ethics of artificial intelligence and federated learning. He co-authored the book "Federated Learning" - the first monograph on the topic of federated learning.

**Xiaofei Xie** received the B.E., M.E., and Ph.D. degrees from Tianjin University. He is currently a Presidential Post-Doctoral Fellow with Nanyang Technological University, Singapore. He has published some top tier conference/journal papers relevant to software analysis in ISSTA, FSE, TSE, IJCAI, and CCS. His main research interests include program analysis, loop analysis, traditional software testing, and security analysis of artificial intelligence. In particular, he won two ACM SIGSOFT Distinguished Paper awards.



**Lei Ma** received the B.E. degree from Shanghai Jiao Tong University, Shanghai, China, and the M.E. and Ph.D. degrees from The University of Tokyo, Tokyo, Japan. He is currently an Associate Professor and Canada CIFAR AI Chair with the University of Alberta, Edmonton, AB, Canada. He also holds a Research Fellow position, co-leading Intelligent Software Engineering Lab, Kyushu University, Fukuoka, Japan, and honorably affiliated with Alberta Machine Intelligence Institute, Edmonton, AB, Canada. His recent research centers around the interdisciplinary fields of software engineering (SE) and trustworthy artificial intelligence (AI) with a special focus on the quality and reliability assurance of machine learning and AI Systems. Many of his works were published in top-tier software engineering and AI venues (e.g., TSE, ICSE, FSE, ASE, ISSTA, ICML, NeurIPS, ACM MM, AAAI, IJCAI, ECCV, and CAV). Dr. Ma is a recipient of more than 10 prestigious academic awards, including three ACM SIGSOFT Distinguished Paper Awards.

**Tao Xiang** received the BEng, MS and PhD degrees in computer science from Chongqing University, China, in 2003, 2005, and 2008, respectively. He is currently a Professor of the College of Computer Science at Chongqing University. Prof. Xiang's research interests include multimedia security, cloud security, data privacy and cryptography. He has published over 100 papers on international journals and conferences. He also served as a referee for numerous international journals and conferences.

**Yang Liu** received the B.Comp. degree (Hons.) from the National University of Singapore (NUS) in 2005 and the Ph.D. degree from NUS and MIT, in 2010. He started his postdoctoral work in NUS and MIT. In 2012, he joined Nanyang Technological University (NTU). He is currently a Full Professor and the Director of the Cybersecurity Laboratory, NTU. He specializes in software verification, security, and software engineering. His research has bridged the gap between the theory and practical usage of formal methods and program analysis to evaluate the design and implementation of software for high assurance and security. By now, he has more than 270 publications in top tier conferences and journals. He received a number of prestigious awards, including the MSRA Fellowship, the TRF Fellowship, the Nanyang Assistant Professor, the Tan Chin Tuan Fellowship, the Nanyang Research Award, and eight best paper awards in top conferences, such as ASE, FSE, and ICSE.