Detection and Mitigation of Byzantine Attacks in Distributed Training

Konstantinos Konstantinidis, Namrata Vaswani, *Fellow, IEEE*,
and Aditya Ramamoorthy, *Senior Member, IEEE*

***Abstract*— A plethora of modern machine learning tasks require the utilization of large-scale distributed clusters as a critical component of the training pipeline. However, abnormal Byzantine behavior of the worker nodes can derail the training and compromise the quality of the inference. Such behavior can be attributed to unintentional system malfunctions or orchestrated attacks; as a result, some nodes may return arbitrary results to the parameter server (PS) that coordinates the training. Recent work considers a wide range of attack models and has explored robust aggregation and/or computational redundancy to correct the distorted gradients. In this work, we consider attack models ranging from strong ones: q omniscient adversaries with full knowledge of the defense protocol that can change from iteration to iteration to weak ones: q randomly chosen adversaries with limited collusion abilities which only change every few iterations at a time. Our algorithms rely on redundant task assignments coupled with detection of adversarial behavior. We also show the convergence of our method to the optimal point under common assumptions and settings considered in literature. For strong attacks, we demonstrate a reduction in the fraction of distorted gradients ranging from 16%–99% as compared to the prior state-of-the-art. Our top-1 classification accuracy results on the CIFAR-10 data set demonstrate 25% advantage in accuracy (averaged over strong and weak scenarios) under the most sophisticated attacks compared to state-of-the-art methods.**

***Index Terms*— Byzantine resilience, distributed training, gradient descent, deep learning, optimization, security.**

I. Introduction and Background

INCREASINGLY complex machine learning models with large data set sizes are nowadays routinely trained on distributed clusters. A typical setup consists of a single central machine (*parameter server* or PS) and multiple worker machines. The PS owns the data set, assigns gradient tasks to workers, and coordinates the protocol. The workers then compute gradients of the loss function with respect to the model parameters. These computations are returned to the PS, which *aggregates* them, updates the model, and maintains the global copy of it. The new copy is communicated back to the workers. Multiple iterations of this process are performed until convergence has been achieved. PyTorch [1], TensorFlow [2], MXNet [3], CNTK [4] and other frameworks support this architecture.

These setups offer significant speedup benefits and enable training challenging, large-scale models. As inference problems scale, such models would be impossible to solve on one machine due to physical limitations on how many resources a single system can be built with (*vertical scaling*); instead, a cluster of servers is utilized (*horizontal scaling*) to jointly execute the overall training task. Nevertheless, the servers are vulnerable to misbehavior by the worker nodes, i.e., when a subset of them returns erroneous computations to the PS, either inadvertently or on purpose. This “*Byzantine*” behavior can be attributed to a wide range of reasons. The principal causes of inadvertent errors are hardware and software malfunctions (e.g., [5]). Reference [6] exposes the vulnerability of neural networks to such failures and identifies weight parameters that could maximize accuracy degradation. The gradients may also be distorted in an adversarial manner. As ML problems demand more resources, many jobs are often outsourced to external commodity servers (cloud) whose security cannot be guaranteed. Thus, an adversary may be able to gain control of some devices and fool the model. The distorted gradients can derail the optimization and lead to low test accuracy or slow convergence.

Achieving robustness in the presence of Byzantine node behavior and devising training algorithms that can efficiently aggregate the gradients has inspired several works [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18]. The first idea is to filter the corrupted computations from the training without attempting to identify the Byzantine workers. Specifically, many existing papers use majority voting and median-based defenses [7], [8], [9], [10], [11], [12], [13] for this purpose. In addition, several works also operate by replicating the gradient tasks [14], [15], [16], [17], [18] allowing for consistency checks across the cluster. The second idea for mitigating Byzantine behavior involves detecting the corrupted devices and subsequently ignoring their calculations [19], [20], [21], in some instances paired with redundancy [17]. In this work, we propose a technique that combines the usage of redundant tasks, filtering, and detection of Byzantine workers. Our work is applicable to a broad range of assumptions on the Byzantine behavior.

There is much variability in the adversarial assumptions that prior work considers. For instance, prior work differs in

Manuscript received 16 August 2022; revised 10 May 2023; accepted 20 September 2023; approved by IEEE/ACM Transactions on Networking Editor G. Joshi. Date of publication 23 October 2023; date of current version 18 April 2024. This work was supported in part by the National Science Foundation (NSF) under Grant CCF-1910840 and Grant CCF-2115200. The material in this work has appeared in part at the 2022 IEEE International Symposium on Information Theory (ISIT) [DOI: 10.1109/ISIT50566.2022.9834813]. *(Corresponding author: Aditya Ramamoorthy.)*

Konstantinos Konstantinidis was with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA. He is now with C3.ai Inc., Redwood City, CA 94063 USA (e-mail: k.m.konstantinidis@gmail.com).

Namrata Vaswani and Aditya Ramamoorthy are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: namrata@iastate.edu; adityar@iastate.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2023.3324697, provided by the authors.

Digital Object Identifier 10.1109/TNET.2023.3324697

the maximum number of adversaries considered, their ability to collude, their possession of knowledge involving the data assignment and the protocol, and whether the adversarial machines are chosen at random or systematically. We will initially examine our methods under strong adversarial models similar to those in prior work [10], [11], [14], [22], [23], [24], [25]. We will then extend our algorithms to tackle weaker failures that are not necessarily adversarial but rather common in commodity machines [5], [6], [26]. We expand on related work in the upcoming Section II.

II. Related Work and Summary of Contributions

A. Related Work

All work in this area (including ours) assumes a reliable parameter server that possesses the global data set and can assign specific subsets of it to workers. *Robust aggregation* methods have also been proposed for federated learning [27], [28]; however, as we make no assumption of privacy, our work, as well as the methods we compare with do not apply to federated learning.

One category of defenses splits the data set into K *batches* and assigns one to each worker with the ultimate goal of suitably aggregating the results from the workers. Early work in the area [12] established that no *linear aggregation* method (such as averaging) can be robust even to a single adversarial worker. This has inspired alternative methods collectively known as *robust aggregation*. Majority voting, geometric median, and squared-distance-based techniques fall into this category [8], [9], [10], [11], [12], [13].

One of the most popular robust aggregation techniques is known as *mean-around-median* or *trimmed mean* [10], [11]. It handles each dimension of the gradient separately and returns the average of a subset of the values that are closest to the median. *Auror* [25] is a variant of trimmed mean which partitions the values of each dimension into two clusters using *k-means* and discards the smaller cluster if the distance between the two exceeds a threshold; the values of the larger cluster are then averaged. *signSGD* in [26] transmits only the sign of the gradient vectors from the workers to the PS and exploits majority voting to decide the overall update; this practice reduces the communication time and denies any individual worker too much effect on the update.

Krum in [12] chooses a single honest worker for the next model update, discarding the data from the rest of them. The chosen gradient is the one closest to its $k \in \mathbb{N}$ nearest neighbors. In later work [24], the authors recognized that Krum may converge to an *ineffectual* model in the landscape of non-convex high dimensional problems, such as in neural networks. They showed that a large adversarial change to a single parameter with a minor impact on the L^p norm can make the model ineffective. In the same work, they present an alternative defense called *Bulyan* to oppose such attacks. The algorithm works in two stages. In the first part, a *selection set* of potentially benign values is iteratively constructed. In the second part, a variant of trimmed mean is applied to the selection set. Nevertheless, if K machines are used, Bulyan is designed to defend only up to $(K-3)/4$ fraction of corrupted workers.

Another category of defenses is based on *redundancy* and seeks resilience to Byzantines by replicating the gradient computations such that each of them is processed by more than one machine [15], [16], [17], [18]. Even though this approach requires more computation load, it comes with stronger guarantees of correcting the erroneous gradients. Existing redundancy-based techniques are sometimes combined with robust aggregation [16]. The main drawback of recent work in this category is that the training can be easily disrupted by a powerful, omniscient adversary that has full control of a subset of the nodes and can mount judicious attacks [14].

Redundancy-based method *DRACO* in [17] uses a simple *Fractional Repetition Code* (FRC) (that operates by grouping workers) and the cyclic repetition code introduced in [29] and [30] to ensure robustness; majority voting and Fourier decoders try to alleviate the adversarial effects. Their work ensures exact recovery (as if the system had no adversaries) with q Byzantine nodes, when each task is replicated $r \geq 2q+1$ times; the bound is information-theoretic minimum, and DRACO is not applicable if it is violated. Nonetheless, this requirement is very restrictive for the typical assumption that up to half of the workers can be Byzantine.

DETOX in [16] extends DRACO and uses a simple grouping strategy to assign the gradients. It performs multiple stages of aggregation to gradually filter the adversarial values. The first stage involves majority voting, while the following stages perform robust aggregation. Unlike DRACO, the authors do not seek exact recovery; hence the minimum requirement in r is small. However, the theoretical resilience guarantees that DETOX provides depend heavily on a "random choice" of the adversarial workers. In fact, we have crafted simple attacks [14] to make this aggregator fail under a more careful choice of adversaries. Furthermore, their theoretical results hold when the fraction of Byzantines is less than $1/40$.

A third category focuses on *ranking* and/or *detection* [17], [19], [20]; the objective is to rank workers using a reputation score to identify suspicious machines and exclude them or give them lower weight in the model update. This is achieved by means of computing reputation scores for each machine or by using ideas from coding theory to assign tasks to workers (encoding) and to detect the adversaries (decoding). *Zeno* in [20] ranks each worker using a score that depends on the estimated loss and the magnitude of the update. Zeno requires strict assumptions on the smoothness of the loss function and the gradient estimates' variance to tolerate an adversarial majority in the cluster. Similarly, *ByGARS* [19] computes reputation scores for the nodes based on an auxiliary data set; these scores are used to weigh the contribution of each gradient to the model update.

B. Contributions

In this paper, we propose novel techniques which combine *redundancy*, *detection*, and *robust aggregation* for Byzantine resilience under a range of attack models and assumptions on the dataset/loss function.

Our first scheme *Aspis* is a subset-based assignment method for allocating tasks to workers in strong adversarial settings: up to q omniscient, colluding adversaries that can change at each iteration. We also consider weaker attacks: adversaries chosen randomly with limited collusion abilities, changing only after a few iterations at a time. It is conceivable that

Aspis should continue to perform well with weaker attacks. However, as discussed later (Section V-B), Aspis requires large batch sizes (for the mini-batch SGD). It is well-recognized that large batch sizes often cause performance degradation in training [31]. Accordingly, for this class of attacks, we present a different algorithm called *Aspis+* that can work with much smaller batch sizes. Both Aspis and Aspis+ use combinatorial ideas to assign the tasks to the worker nodes. Our work builds on our initial work in [22] and makes the following contributions.

- We demonstrate a worst-case upper bound (under any possible attack) on the fraction of corrupted gradients when Aspis is used. Even in this adverse scenario, our method enjoys a reduction in the fraction of corrupted gradients of more than 90% compared with DETOX [16]. A weaker variation of this attack is where the adversaries do not collude and act randomly. In this case, we demonstrate that the Aspis protocol allows for detecting all the adversaries. In both scenarios, we provide theoretical guarantees on the fraction of corrupted gradients.
- In the setting where the dataset is distributed i.i.d. and the loss function is strongly convex and other technical conditions hold, we demonstrate a proof of convergence for Aspis. We demonstrate numerical results on the linear regression problem in this part; these show the advantage of Aspis over competing methods such as DETOX.
- For weaker attacks (discussed above), our experimental results indicate that Aspis+ detects all adversaries within approximately 5 iterations.
- We present top-1 classification accuracy experiments on the CIFAR-10 [32] data set for various gradient distortion attacks coupled with choice/behavior patterns of the adversarial nodes. Under the most sophisticated distortion methods [23], the performance gap between Aspis/Aspis+ and other state-of-the-art methods is substantial, e.g., for Aspis it is 43% in the strong scenario (*cf.* Figure 7a), and for Aspis+ 19% in the weak scenario (*cf.* Figure 13).

III. Distributed Training Formulation

Assume a loss function $l_i(\mathbf{w})$ for the i^{th} sample of the dataset where $\mathbf{w} \in \mathbb{R}^d$ is the set of parameters of the model.[1] The objective of distributed training is to minimize the empirical loss function $\hat{L}(\mathbf{w})$ with respect to $\mathbf{w}$, where

$$\hat{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} l_i(\mathbf{w}).$$

Here n denotes the number of samples.

We use either gradient descent (GD) or mini-batch Stochastic Gradient Descent (SGD) to solve this optimization. In both methods, initially $\mathbf{w}$ is randomly set to $\mathbf{w}_0$ ($\mathbf{w}_t$ is the model state at the end of iteration t). When using GD, the update equation is

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \frac{1}{n}\sum_{i=1}^{n} \nabla l_i(\mathbf{w}_t). \quad (1)$$

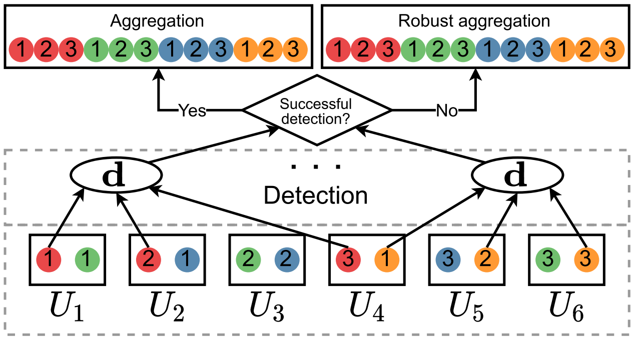


Fig. 1. Aggregation of gradients on a cluster.

Under mini-batch SGD a random *batch* B_t of b samples is chosen to perform the update in the t^{th} iteration. Thus,

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \frac{1}{|B_t|}\sum_{i \in B_t} \nabla l_i(\mathbf{w}_t). \quad (2)$$

In both methods η_t is the learning rate at the t^{th} iteration. The workers denoted $U_1, U_2, \ldots, U_K$, compute gradients on subsets of the batch. The training is *synchronous*, i.e., the PS waits for all workers to return before performing an update. It stores the data set and the model and coordinates the protocol. It can be observed that GD can be considered an instance of mini-batch SGD where the batch at each iteration is the entire dataset. Our discussion below is in the context of mini-batch SGD but can easily be applied to the GD case by using this observation.

We consider settings in this work that depend on the underlying assumptions on the dataset and the loss function. Setting-I does not make any assumption on the dataset or the loss function. In Setting-II at the top-level (technical details appear in Section VI) we assume that the data samples are distributed i.i.d. and the loss function is strongly-convex. The results that we provide depend on the underlying setting.

Task assignment: Each batch B_t is split into f disjoint subsets $\{B_{t,i}\}_{i=0}^{f-1}$, which are then assigned to the workers according to our placement policy. In what follows we refer to these as "files" to avoid confusion with other subsets that we need to refer to. Computational redundancy is introduced by assigning a given file to $r > 1$ workers. As the load on all the workers is equal it follows that each worker is responsible for $l = fr/K$ files (l is the *computation load*). We let $\mathcal{N}^w(U_j)$ be the set of files assigned to worker U_j and $\mathcal{N}^f(B_{t,i})$ be the group of workers assigned to file $B_{t,i}$; our placement scheme is such that $\mathcal{N}^f(B_{t,i})$ uniquely identifies the file $B_{t,i}$; thus, we will sometimes refer to the file $B_{t,i}$ by its worker assignment, $\mathcal{N}^f(B_{t,i})$. We will also occasionally use the term *group* (of the assigned workers) to refer to a file. We discuss the actual placement algorithms used in this work in the upcoming subsection III-A.

Training: Each worker U_j is given the task of computing the sum of the gradients on all its assigned files. For example, if file $B_{t,i}$ is assigned to U_j, then it calculates $\sum_{i' \in B_{t,i}} \nabla l_{i'}(\mathbf{w}_t)$ and returns them to the PS. In every iteration, the PS will run our detection algorithm once it receives the results from all the users in an effort to identify the q adversaries and will act according to the detection outcome.

[1] The paper's heavily-used notation is summarized in Appendix Table II.

Figure 1 depicts this process. There are $K = 6$ machines and $f = 4$ distinct files (represented by colored circles) replicated $r = 3$ times.[2] Each worker is assigned to $l = 2$ files and computes the sum of gradients (or a distorted value) on each of them. The "**d**" ellipses refer to PS's detection operations immediately after receiving all the gradients.

Metrics: We consider various metrics in our work. For Setting-I we consider (i) the fraction of distorted files, and (ii) the top-1 test accuracy of the final trained model. For the distortion fraction, let us denote the number of distorted files upon detection and aggregation by $c^{(q)}$ and its maximum value (under a worst-case attack) by $c^{(q)}_{\max}$. The *distortion fraction* is $\epsilon := c^{(q)}/f$. The top-1 test accuracy is determined via numerical experiments. In Setting-II, in addition we consider proofs and rates of convergence of the proposed algorithms. We provide theoretical results and supporting experimental results on these.

A. Task Assignment

Let $\mathcal{U}$ be the set of workers. Our scheme has $|\mathcal{U}| \leq f$ (i.e., fewer workers than files). Our assignment of files to worker nodes is specified by a bipartite graph $\mathbf{G}_{task}$ where the left vertices correspond to the workers, and the right vertices correspond to the files. An edge in $\mathbf{G}_{task}$ between worker U_i and a file $B_{t,j}$ indicates that the U_i is responsible for processing file $B_{t,j}$.

1) Aspis: For the Aspis scheme we construct $\mathbf{G}_{task}$ as follows. The left vertex set is $\{1, 2, \ldots, K\}$ and the right vertex set corresponds to r-sized subsets of $\{1, 2, \ldots, K\}$ (there are $\binom{K}{r}$ of them). An edge between $1 \leq i \leq K$ and $S \subset \{1, 2, \ldots, K\}$ (where $|S| = r$) exists if $i \in S$. The worker set $\{U_1, \ldots, U_K\}$ is in one-to-one correspondence with $\{1, 2, \ldots, K\}$ and the files $B_{t,0}, \ldots, B_{t,f-1}$ are in one-to-one correspondence with the r-sized subsets.

Example 1: Consider $K = 7$ workers $U_1, U_2 \ldots, U_7$ and $r = 3$. Based on our protocol, the $f = \binom{7}{3} = 35$ files of each batch B_t are associated one-to-one with 3-subsets of $\mathcal{U}$, e.g., the subset $S = \{U_1, U_2, U_3\}$ corresponds to file $B_{t,0}$ and will be processed by U_1, U_2, and U_3.

Remark 1: Our task assignment ensures that every pair of workers processes $\binom{K-2}{r-2}$ files. Moreover, the number of adversaries is $q < K/2$. Thus, upon receiving the gradients from the workers, the PS can examine them for consistency and flag certain nodes as adversarial if their computed gradients differ from $q + 1$ or more of the other nodes. We use this intuition to detect and mitigate the adversarial effects and compute the fraction of corrupted files.

2) Aspis+: For Aspis+, we use combinatorial designs [33] to assign the gradient tasks to workers. Formally, a *design* is a pair $(X, \mathcal{A})$ consisting of a set of v elements (*points*), X, and a family $\mathcal{A}$ (i.e., multiset) of nonempty subsets of X called *blocks*, where each block has the same cardinality k. Similar to Aspis, the workers and files are in one-to-one correspondence with the points and the blocks, respectively. Hence, for our purposes, the k parameter of the design is the redundancy. A $t - (v, k, \lambda)$ design is one where any subset of t points appear together in exactly λ blocks. The case of $t = 2$ has been studied extensively in the literature and is referred to as a *balanced incomplete block design* (BIBD). A bipartite graph representing the incidence between the points and the blocks can be obtained naturally by letting the points correspond to the left vertices, and the blocks correspond to the right vertices. An edge exists between a point and a block if the point is contained in the block.

Example 2: A $2 - (7, 3, 1)$ design, also known as the *Fano plane*, consists of the $v = 7$ points $X = \{1, 2, \ldots, 7\}$ and the block multiset $\mathcal{A}$ contains the blocks $\{1, 2, 3\}$, $\{1, 4, 7\}$, $\{2, 4, 6\}$, $\{3, 4, 5\}$, $\{2, 5, 7\}$, $\{1, 5, 6\}$ and $\{3, 6, 7\}$ with each block being of size $k = 3$. In the bipartite graph $\mathbf{G}_{task}$ representation, we would have an edge, e.g., between point 2 and blocks $\{1, 2, 3\}, \{2, 4, 6\}$, and $\{2, 5, 7\}$.
In Aspis+ we construct $\mathbf{G}_{task}$ by the bipartite graph representing an appropriate $2 - (v, k, \lambda)$ design.

Another change compared to the Aspis placement scheme is that the points of the design will be randomly permuted at each iteration, i.e., for permutation π, the PS will map $\{U_1, U_2, \ldots, U_K\} \xrightarrow{\pi} \{\pi(U_1), \pi(U_2), \ldots, \pi(U_K)\}$. For instance, let us circularly permute the points of the Fano plane in Example 2 as $\pi(U_i) = U_{i+1}, i = 1, 2, \ldots, K-1$ and $\pi(U_K) = U_1$. Then, the file assignment at the next iteration will be based on the block collection $\mathcal{A} = \{\{2, 3, 4\}, \{1, 2, 5\}, \{3, 5, 7\}, \{4, 5, 6\}, \{1, 3, 6\}, \{2, 6, 7\}, \{1, 4, 7\}\}$. Permuting the assignment causes each Byzantine to disagree with more workers and to be detected in fewer iterations; details will be discussed in Section V-C. Owing to this permutation, we use a time subscript for the files assigned to U_i for the t^{th} iteration; this is denoted by $\mathcal{N}^w_t(U_i)$.

IV. Adversarial Attack Models and Gradient Distortion Methods

We now discuss the different Byzantine models that we consider in this work. For all the models, we assume that at most $q < K/2$ workers can be adversarial. For each assigned file $B_{t,i}$ a worker U_j will return the value $\hat{\mathbf{g}}^{(j)}_{t,i}$ to the PS. Then,

$$\hat{\mathbf{g}}^{(j)}_{t,i} = \begin{cases} \mathbf{g}_{t,i} & \text{if } U_j \text{ is honest,} \\ * & \text{otherwise,} \end{cases} \tag{3}$$

where $\mathbf{g}_{t,i}$ is the sum of the loss gradients on all samples in file $B_{t,i}$, i.e.,

$$\mathbf{g}_{t,i} = \sum_{j \in B_{t,i}} \nabla l_j(\mathbf{w}_t)$$

and $*$ is any arbitrary vector in $\mathbb{R}^d$. Within this setup, we examine adversarial scenarios that differ based on the behavior of the workers. Table I provides a high-level summary of the Byzantine models considered in this work as well as in related papers. As we will discuss in Section VIII-B, for those schemes that do not involve redundancy and merely split the work equally among the K workers, all possible choices of the Byzantine set are equivalent, and no *orchestration*[3] of them will change the defense's output; hence, those cases are denoted by "N/A" in the table.

[2] Some arrows and ellipses have been omitted from Figure 1; however, all files will be going through detection.

[3] We will use the term *orchestration* to refer to the method adversaries use to collude and attack collectively as a group.

TABLE I
ADVERSARIAL MODELS CONSIDERED IN LITERATURE

Scheme	Byzantine choice/orchestration	Gradient distortion
Draco [17]	optimal	reversed gradient, constant
DETOX [16]	random	ALIE, reversed gradient, constant
ByzShield [14]	optimal	ALIE, reversed gradient, constant
Bulyan [24]	N/A	ℓ_2-norm attack targeted on Bulyan
Multi-Krum [12]	N/A	random high-variance Gaussian vector
Aspis	ATT-1, ATT-2	ALIE, FoE, reversed gradient
Aspis+	ATT-3	ALIE, constant

A. Attack 1

We first consider a weak attack, denoted ATT-1, where the Byzantine nodes operate independently (i.e., do not collude) and attempt to distort the gradient on any file they participate in. For instance, a node may try to return arbitrary gradients on all its assigned files. For this attack, the identity of the workers may be arbitrary at each iteration as long as there are at most q of them.

Remark 2: We emphasize that even though we call this attack "weak", this is the attack model considered in several prior works [16], [17]. To our best knowledge, most of them have not considered the adversarial problem from the lens of detection.

B. Attack 2

Our second scenario, named ATT-2, is the strongest one we consider. We assume that the adversaries have full knowledge of the task assignment at each iteration and the detection strategies employed by the PS. The adversaries can collude in the "best" possible way to corrupt as many gradients as possible. Moreover, the set of adversaries can also change from iteration to iteration as long as there are at most q of them.

C. Attack 3

This attack is similar to ATT-1 and will be called ATT-3. On the one hand, it is weaker in the sense that the set of Byzantines (denoted A) does not change in every iteration. Instead, we will assume that there is a "Byzantine window" of T_b iterations in which the set A remains fixed. Also, the set A will be a randomly chosen set of q workers from $\mathcal{U}$, i.e., it will not be chosen systematically. A new set will be chosen at random at all iterations t, where $t \equiv 0 \pmod{T_b}$. Conversely, it is stronger than ATT-1 since we allow for limited collusion amongst the adversarial nodes. In particular, the Byzantines simulated by ATT-3 will distort only the files for which a Byzantine majority exists.

D. Gradient Distortion Methods

For each of the attacks considered above, the adversaries can distort the gradient in specific ways. Several such techniques have been considered in the literature and our numerical experiments use these methods for comparing different methods. For instance, *ALIE* [23] involves communication among the Byzantines in which they jointly estimate the mean μ_i and standard deviation σ_i of the batch's gradient for each dimension i and subsequently use them to construct a distorted gradient that attempts to distort the median of the results. Another powerful attack is *Fall of Empires (FoE)* [34] which performs "inner product manipulation" to make the inner product between the true gradient and the robust estimator to be negative even when their distance is upper bounded by a small value. *Reversed gradient* distortion returns $-cg$ for $c > 0$, to the PS instead of the true gradient g. The *constant attack* involves the Byzantine workers sending a constant gradient with all elements equal to a fixed value. To our best knowledge, the ALIE algorithm is the most sophisticated attack in literature for deep learning techniques.

V. Defense Strategies in Aspis and Aspis+

In our work we use the Aspis task assignment and detection strategy for attacks ATT-1 and ATT-2. For ATT-3, we will use Aspis+. Recall that the methods differ in their corresponding task assignments. Nevertheless, the central idea in both detection methods is for the PS to apply a set of consistency checks on the obtained gradients from the different workers at each iteration to identify the adversaries.

Let the current set of adversaries be $A \subset \{U_1, U_2, \ldots, U_K\}$ with $|A| = q$; also, let H be the honest worker set. The set A is unknown, but our goal is to provide an estimate $\hat{A}$ of it. Ideally, the two sets should be identical. In general, depending on the adversarial behavior, we will be able to provide a set $\hat{A}$ such that $\hat{A} \subseteq A$. For each file, there is a group of r workers which have processed it, and there are $\binom{r}{2}$ pairs of workers in each group. Each such pair may or may not agree on the gradient value for the file. For iteration t, let us encode the agreement of workers U_{j_1}, U_{j_2} on common file i during the current iteration t by

$$\alpha_{t,i}^{(j_1,j_2)} := \begin{cases} 1 & \text{if } \hat{\mathbf{g}}_{t,i}^{(j_1)} = \hat{\mathbf{g}}_{t,i}^{(j_2)}, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Across all files, the total number of agreements between a pair of workers U_{j_1}, U_{j_2} during the t^{th} iteration is denoted by

$$\alpha_t^{(j_1,j_2)} := \sum_{i \in \mathcal{N}_t^w(U_{j_1}) \cap \mathcal{N}_t^w(U_{j_2})} \alpha_{t,i}^{(j_1,j_2)}. \quad (5)$$

Since the placement is known, the PS can always perform the above computation. Next, we form an undirected graph $\mathbf{G}_t$ whose vertices correspond to all workers $\{U_1, U_2, \ldots, U_K\}$. An edge (U_{j_1}, U_{j_2}) exists in $\mathbf{G}_t$ only if the computed gradients (at iteration t) of U_{j_1} and U_{j_2} match in "all" their common assignments.

A. Aspis Detection Rule

In what follows, we suppress the iteration index t since the Aspis algorithm is the same for each iteration. For the Aspis

Algorithm 1 Proposed Aspis Graph-Based Detection

```
Input: Computed gradients $\hat{\mathbf{g}}_{t,i}^{(j)}$, $i = 0, 1, \ldots, f-1$,
       $j = 1, 2, \ldots, K$, redundancy $r$ and empty
       graph $\mathbf{G}$ with worker vertices $\mathcal{U}$.
for each pair $(U_{j_1}, U_{j_2}), j_1 \neq j_2$ of workers do
    PS computes the number of agreements $\alpha^{(j_1,j_2)}$ of
      the pair $U_{j_1}, U_{j_2}$ on the gradient value.
    if $\alpha^{(j_1,j_2)} = \binom{K-2}{r-2}$ then
        Connect vertex $U_{j_1}$ to vertex $U_{j_2}$ in $\mathbf{G}$.
    end
end
PS enumerates all $k$ maximum cliques
  $M_{\mathbf{G}}^{(1)}, M_{\mathbf{G}}^{(2)}, \ldots, M_{\mathbf{G}}^{(k)}$ in $\mathbf{G}$.
if there is a unique maximum clique $M_{\mathbf{G}}$ ($k = 1$) then
    PS determines the honest workers $H = M_{\mathbf{G}}$ and
      the adversarial machines $\hat{A} = \mathcal{U} - M_{\mathbf{G}}$.
else
    PS declares unsuccessful detection.
end
```

Algorithm 2 Proposed Aspis/Aspis+ Aggregation Protocol to Alleviate Byzantine Effects

```
Input: Data set of $n$ samples, batch size $b$,
       computation load $l$, redundancy $r$, number of
       files $f$, maximum iterations $T$, file assignments
       $\{\mathcal{N}^w(U_i)\}_{i=1}^K$, robust estimator function $\widehat{\text{med}}$.
The PS randomly initializes model's parameters to $\mathbf{w}_0$.
for $t = 0$ to $T-1$ do
    PS chooses a random batch $B_t \subseteq \{1, 2, \ldots, n\}$ of
      $b$ samples, partitions it into $f$ files $\{B_{t,i}\}_{i=0}^{f-1}$ and
      assigns them to workers according to
      $\{\mathcal{N}^w(U_i)\}_{i=1}^K$. It then transmits $\mathbf{w}_t$ to all workers.
    for each worker $U_j$ do
        if $U_j$ is honest then
            for each file $i \in \mathcal{N}^w(U_j)$ do
                $U_j$ computes the sum of gradients
                    $\hat{\mathbf{g}}_{t,i}^{(j)} = \sum_{k \in B_{t,i}} \nabla l_k(\mathbf{w}_t)$.
            end
        else
            $U_j$ constructs $l$ adversarial vectors
                $\hat{\mathbf{g}}_{t,i_1}^{(j)}, \hat{\mathbf{g}}_{t,i_2}^{(j)}, \ldots, \hat{\mathbf{g}}_{t,i_l}^{(j)}$.
        end
        $U_j$ returns $\hat{\mathbf{g}}_{t,i_1}^{(j)}, \hat{\mathbf{g}}_{t,i_2}^{(j)}, \ldots, \hat{\mathbf{g}}_{t,i_l}^{(j)}$ to the PS.
    end
    PS runs a detection algorithm to identify the
      adversaries.
    if detection is successful then
        Let $H$ be the detected honest workers. Initialize
          a non-corrupted gradient set as $\mathcal{G} = \emptyset$.
        for each file in $\{B_{t,i}\}_{i=0}^{f-1}$ do
            PS chooses the gradient of a worker in
              $\mathcal{N}^f(B_{t,i}) \cap H$ (if non-empty) and adds it
              to $\mathcal{G}$.
        end
            $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \frac{1}{|\mathcal{G}|} \sum_{\mathbf{g} \in \mathcal{G}} \mathbf{g}$.
    else
        for each file in $\{B_{t,i}\}_{i=0}^{f-1}$ do
            PS determines the $r$ workers in $\mathcal{N}^f(B_{t,i})$
              which have processed $B_{t,i}$ and computes
              $\mathbf{m}_i = \text{majority}\left\{\hat{\mathbf{g}}_{t,i}^{(j)} : U_j \in \mathcal{N}^f(B_{t,i})\right\}$.
        end
        PS updates the model via
          $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \times \widehat{\text{med}}\{\mathbf{m}_i : i = 0, 1, \ldots, f-1\}$.
    end
end
```

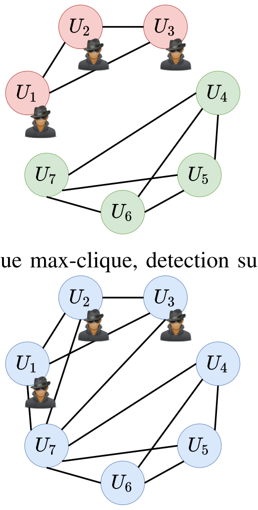
task assignment (*cf.* Section III-A.1), any two workers, U_{j_1} and U_{j_2}, have $\binom{K-2}{r-2}$ common files.

Let us index the q adversaries in $A = \{A_1, A_2, \ldots, A_q\}$ and the honest workers in H. We say that two workers U_{j_1} and U_{j_2} disagree if there is no edge between them in $\mathbf{G}$. The non-existence of an edge between U_{j_1} and U_{j_2} only means that they disagree in *at least one* of the $\binom{K-2}{r-2}$ files that they jointly participate in. For corrupting the gradients, each adversary has to disagree on the computations with a subset of the honest workers. An adversary may also disagree with other adversaries.

A *clique* in an undirected graph is defined as a subset of vertices with an edge between any pair of them. A *maximal clique* is one that cannot be enlarged by adding additional vertices to it. A *maximum clique* is one such that there is no clique with more vertices in the given graph. We note that the set of honest workers H will pair-wise agree on all common tasks. Thus, H forms a clique (of size $K - q$) within $\mathbf{G}$. The clique containing the honest workers may not be maximal. However, it will have a size of at least $K - q$. Let the maximum clique on $\mathbf{G}$ be $M_{\mathbf{G}}$. Any worker U_j with $\deg(U_j) < K - q - 1$ will not belong to a maximum clique and can right away be eliminated as a "detected" adversary.

The essential idea of our detection is to run a *clique-finding* algorithm on $\mathbf{G}$ (summarized in Algorithm 1). The detection may be successful or unsuccessful depending on which attack is used; we discuss this in more detail shortly.

We note that clique-finding is well-known to be an NP-complete problem [35]. Nevertheless, there are fast, practical algorithms with excellent performance on graphs even up to hundreds of nodes [36], [37]. Specifically, the authors of [37] have shown that their proposed algorithm, which enumerates all maximal cliques, has similar complexity as other methods [38], [39], which are used to find a single maximum clique. We utilize this algorithm. Our extensive experimental evidence suggests that clique-finding is not a computation bottleneck for the size and structure of the graphs that Aspis uses. We have experimented with clique-finding on a graph of $K = 100$ workers and $r = 5$ for different values of q; in all cases, enumerating all maximal cliques took no more than 15 milliseconds. These experiments and the asymptotic complexity of the entire protocol are addressed in Supplement Section XI-B.



(a) Unique max-clique, detection succeeds.

(b) Two max-cliques, detection fails.

Fig. 2. Detection graph $\mathbf{G}$ for $K = 7$ workers among which U_1, U_2 and U_3 are the adversaries.

During aggregation (see Algorithm 2), the PS will perform a majority vote across the computations of each file (implementation details in Supplement Section XI-C). Recall that r workers have processed each file. For each such file $B_{t,i}$, the PS decides a majority value $\mathbf{m}_i$

$$\mathbf{m}_i := \text{majority}\left\{\hat{\mathbf{g}}_{t,i}^{(j)} : U_j \in \mathcal{N}^f(B_{t,i})\right\}. \tag{6}$$

Assume that r is odd and let $r' = \frac{r+1}{2}$. Under the rule in Eq. (6), the gradient on a file is distorted only if at least r' of the computations are performed by Byzantines. Following the majority vote, we will further filter the gradients using a robust estimator $\widehat{\text{med}}$ (see Algorithm 2, line 25). This robust estimator is either the coordinate-wise median or the geometric median; a similar setup was considered in [14] and [16]. For example, in Figure 1, all returned values for the red file will be evaluated by a majority vote function on the PS, which decides a single output value; a similar voting is done for the other 3 files. After the voting process, Aspis applies the robust estimator $\widehat{\text{med}}$ on the "winning" gradients $\mathbf{m}_i$, $i = 0, 1, \ldots, f-1$.

1) Defense Strategy Against ATT-1: Under ATT-1, it is clear that a Byzantine node will disagree with at least $K-q$ honest nodes (as, by assumption in Section IV-A, it will disagree with all of them), and thus, the degree of the node in $\mathbf{G}$ will be at most $q-1 < K-q-1$, and it will not be part of the maximum clique. Thus, each of the adversaries will be detected, and their returned gradients will not be considered further. The algorithm declares the (unique) maximum clique as honest and proceeds to aggregation. In particular, assume that h workers $U_{i_1}, U_{i_2}, \ldots, U_{i_h}$ have been identified as honest. For each of the f files, if at least one honest worker processed it, the PS will pick one of the "honest" gradient values. The chosen gradients are then averaged for the update (*cf.* Eq. (2)). For instance, in Figure 1, assume that U_1, U_2, and U_4 have been identified as faulty. During aggregation, the PS will ignore the red file as all 3 copies have been compromised. For the orange file, it will pick either the gradient computed by U_5 or U_6 as both of them are "honest." The only files that can be distorted in this case are those that consist exclusively of adversarial nodes.

Figure 2a (corresponding to Example 1) shows an example where in a cluster of size $K = 7$, the $q = 3$ adversaries are $A = \{U_1, U_2, U_3\}$ and the remaining workers are honest with $H = \{U_4, U_5, U_6, U_7\}$. In this case, the unique maximum clique is $M_{\mathbf{G}} = H$, and detection is successful. Under this attack, the distorted files are those whose all copies have been compromised, i.e., $c^{(q)} = \binom{q}{r}$.

2) Defense Strategy Against ATT-2 (Robust Aggregation): Let D_i denote the set of disagreement workers for adversary $A_i, i = 1, 2, \ldots, q$, where D_i can contain members from A and from H. If the attack ATT-2 is used on Aspis, upon the formation of $\mathbf{G}$ we know that a worker U_j will be flagged as adversarial if $deg(U_j) < K-q-1$. Therefore to avoid detection, a *necessary* condition is that $|D_j| \leq q$.

We now upper bound the number of files that can be corrupted under *any possible strategy* employed by the adversaries. Note that according to Algorithm 2, we resort to robust aggregation in case of more than one maximum clique in $\mathbf{G}$. In this scenario, a gradient can only be corrupted if a majority of the assigned workers computing it are adversarial and agree on a wrong value. The proof of the following theorem appears in Appendix Section IX-A.

Theorem 1: Consider a training cluster of K workers with q adversaries using algorithm in Section III-A.1 to assign the $f = \binom{K}{r}$ files to workers, and Algorithm 1 for adversary detection. Under any adversarial strategy, the maximum number of files that can be corrupted is

$$c_{\max}^{(q)} = \frac{1}{2}\binom{2q}{r}. \tag{7}$$

Furthermore, this upper bound can be achieved if all adversaries fix a set $D \subset H$ of honest workers with which they will consistently disagree on the gradient (by distorting it).

Remark 3: We emphasize that the maximum fraction of corrupted gradients $c_{\max}^{(q)}/f$ is much lesser as compared to the baseline q/K and with respect to other schemes as well (details in Sec. VII). For instance $K = 15$ and $q = 3$, at most 0.022 fraction of the gradients are corrupted in Aspis as against 0.2 for the baseline scheme.

In Appendix Section IX-A we show that under ATT-2 there is bound to be more than one maximum clique in the detection graph. Thus, the PS cannot unambiguously decide which one is the honest one; detection fails and we fall back to the robust aggregation technique.

An example is shown in Figure 2 for the setup of Example 1. The adversaries $A = \{U_1, U_2, U_3\}$ consistently disagree with the workers in $D = \{U_4, U_5, U_6\} \subset H$. The ambiguity as to which of the two maximum cliques ($\{U_1, U_2, U_3, U_7\}$ or $\{U_4, U_5, U_6, U_7\}$) is the honest one makes an accurate detection impossible; robust aggregation will be performed instead.

B. Motivation for Aspis+

Our motivation for proposing Aspis+ originates in the limitations of the subset assignment of Aspis. It is evident from the experimental results in Section VIII-B that Aspis is more suitable to worst-case attacks where the adversaries collude and distort the maximum number of tasks in an undetected

Algorithm 3 Proposed Aspis+ Graph-Based Detection

```
Input: Computed gradients $\hat{\mathbf{g}}_{t,i}^{(j)}$, $i = 0, 1, \dots, f-1$,
       $j = 1, 2, \dots, K$, $2-(v, k, \lambda)$ design, length of
       detection window $T_d$, maximum iterations $T$.
for $t = 0$ to $T - 1$ do
    Let $t' = t \pmod{T_d} + 1$.
    if $t' = 1$ then
        Set $\mathbf{G}$ as the complete graph with worker
          vertices $\mathcal{U}$.
        $\forall j_1, j_2$, set $\alpha^{(j_1,j_2)} = 0$.
    end
    for each pair $(U_{j_1}, U_{j_2}), j_1 \neq j_2$ of workers do
        PS computes the number of agreements $\alpha_t^{(j_1,j_2)}$
          of the pair $U_{j_1}, U_{j_2}$ on the gradient value.
        Update $\alpha^{(j_1,j_2)} = \alpha^{(j_1,j_2)} + \alpha_t^{(j_1,j_2)}$.
    end
    for each pair $(U_{j_1}, U_{j_2}), j_1 \neq j_2$ of workers do
        if $\alpha^{(j_1,j_2)} < \lambda \times t'$ then
            Remove edge $(U_{j_1}, U_{j_2})$ from $\mathbf{G}$.
        end
    end
    for each worker $U_j \in \mathcal{U}$ do
        if $deg(U_j) < K - q - 1$ then
            $\hat{A} = \hat{A} \cup \{U_j\}$.
        end
    end
    if $|\hat{A}| > q$ then
        Set $\hat{A}$ to be the $q$ most recently detected
          Byzantines.
    end
end
```

fashion; in this case, the accuracy gap between Aspis and prior methods is maximal. Aspis does not perform as well under weaker attacks such as the *reversed gradient* attack (*cf.* Figures 8a, 8b, 8c even though it achieves a much smaller distortion fraction ϵ, as discussed in Section VII. This can be attributed to the fact that the number of tasks is $\binom{K}{r}$ and even for the considered cluster of $K = 15$, $r = 3$, it would require splitting the batch into 455 files; hence, the batch size must be a multiple of 455. There is significant evidence that large batch sizes can hurt generalization and make the model converge slowly [31], [40], [41]. Some workarounds have been proposed to solve this problem. For instance, the work of [41] uses layer-wise adaptive rate scaling to update each layer using a different learning rate. The authors of [42] perform implicit regularization using warmup and cosine annealing to tune the learning rate as well as gradient clipping. However, these methods require training for a significantly larger number of epochs. For the above reasons, we have extended our work and proposed Aspis+ to handle weaker Byzantine failures (*cf.* ATT-3) while requiring a much smaller batch size.

C. Aspis+ Detection Rule

The principal intuition of the Aspis+ detection approach (used for ATT-3) is to iteratively keep refining the graph $\mathbf{G}$ in which the edges encode the agreements of workers during consecutive and non-overlapping windows of T_d iterations. At the beginning of each such window, the PS will reset $\mathbf{G}$ to be a complete graph, i.e., as if all workers pairwise agree with other. Then, it will gradually remove edges from $\mathbf{G}$ as disagreements between the workers are observed; hence, the graph will be updated at each of the T_d iterations of the window, and the PS will assume that the Byzantine set does not change within a detection window. In practice, as we do not know the "Byzantine window," we will not assume an alignment between the two kinds of windows, and we will set $T_d \neq T_b$ for our experiments. The detection method will be the same for all detection windows; thus, we will analyze the process in one window of T_d steps.

For a detection window, let us encode the agreement of workers U_{j_1}, U_{j_2} on common file i during the current iteration t of the window $t = 1, 2, \dots, T_d$ as

$$\alpha_{t,i}^{(j_1,j_2)} := \begin{cases} 1 & \text{if } \hat{\mathbf{g}}_{t,i}^{(j_1)} = \hat{\mathbf{g}}_{t,i}^{(j_2)}, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Across all files, the total number of agreements between a pair of workers U_{j_1}, U_{j_2} during the t^{th} iteration is denoted by

$$\alpha_t^{(j_1,j_2)} := \sum_{i \in \mathcal{N}_t^w(U_{j_1}) \cap \mathcal{N}_t^w(U_{j_2})} \alpha_{t,i}^{(j_1,j_2)}. \tag{9}$$

Assume that the current iteration of the window is indexed with $t' \in \{1, 2, \dots, T_d\}$. The PS will collect all agreements for each pair of workers U_{j_1}, U_{j_2} up until the current iteration as

$$\alpha^{(j_1,j_2)} := \sum_{t=1}^{t'} \alpha_t^{(j_1,j_2)}. \tag{10}$$

Since the placement is known, the PS can always perform the above computation. Next, it will examine the agreements and update $\mathbf{G}$ as necessary.

Based on the task placement (*cf.* Section III-A.2), an edge (U_{j_1}, U_{j_2}) exists in $\mathbf{G}$ only if the computed gradients of U_{j_1} and U_{j_2} match in all their λ common groups in all iterations up to the current one indexed with t', i.e., a pair U_{j_1}, U_{j_2} needs to have $\alpha^{(j_1,j_2)} = \lambda \times t'$ for an edge (U_{j_1}, U_{j_2}) to be in $\mathbf{G}$. If this is not the case, the edge (U_{j_1}, U_{j_2}) will be removed from $\mathbf{G}$. After all such edges are examined, detection is done using degree counting. Given that there are q Byzantines in the cluster, after examining all pairs of workers and determining the form of $\mathbf{G}$, a worker U_j will be flagged as Byzantine if $deg(U_j) < K - q - 1$. Based on Eq. (10), it is not hard to see that such workers can be eliminated and their gradients will not be considered again until the last iteration of the current window. The only exception to this is if the Byzantine set changes before the end of the current detection window. This is possible due to a potential misalignment between the "Byzantine window" and the detection window (recall that $T_d \neq T_b$ is assumed to avoid trivialities). In this case, more than q workers may be detected as Byzantines; the PS will, by convention, choose $\hat{A}$ to be the most recently detected Byzantines. Algorithm 3 discusses the detection protocol. Following detection, the PS will act as follows. If at least one Byzantine has been detected, it will ignore the votes of detected Byzantines, and for each group, if there is at least one “honest” vote, it will use this as the output of the majority voting group; also, if a group consists merely of detected Byzantines, it will completely ignore the group. The remaining groups will go through robust aggregation (as in Section V-A). In our experiments in Section VIII-C, all Byzantines are detected successfully in at most 5 iterations. Example 3 showcases the utility of permutations in our detection algorithm using $K = 7$ workers.

Example 3: We will use the assignment of Example 2 with $K = 7$ workers $\mathcal{U} = \{1, 2, \ldots, 7\}$ assigned to tasks according to a $2-(7, 3, 1)$ Fano plane and let us denote the assignment of workers to groups (blocks of the design) during the t^{th} iteration by $\mathcal{A}_t$, initially equal to $\mathcal{A}_1 = \{\{1, 2, 3\}, \{1, 4, 7\}, \{2, 4, 6\}, \{3, 4, 5\}, \{2, 5, 7\}, \{1, 5, 6\}, \{3, 6, 7\}\}$. For the windows , assume that $T_d > 2$ and $T_b > 2$. Also , let $q = 2$ and the Byzantine set be $A = \{U_1, U_2\}$. Based on ATT-3 , workers U_1, U_2 are in majority within a group in which they disagree with worker U_3. After the first permutation , a possible assignment is $\mathcal{A}_2 = \{\{1, 3, 6\}, \{3, 4, 7\}, \{2, 4, 6\}, \{1, 4, 5\}, \{5, 6, 7\}, \{2, 3, 5\}, \{1, 2, 7\}\}$. Then , U_1, U_2 are in the same group as the honest U_7 with which they disagree; hence , $deg(U_1) = deg(U_2) = 4 = K - q - 1$, and none of them can afford to disagree with more honest workers to remain undetected. However , if the next permutation assigns the workers as $\mathcal{A}_3 = \{\{1, 3, 6\}, \{1, 4, 7\}, \{4, 6, 5\}, \{2, 3, 4\}, \{2, 6, 7\}, \{1, 2, 5\}, \{3, 5, 7\}\}$ then the adversaries will cast a different vote than U_5 as well. Thus , both of them will be detected after only three iterations.

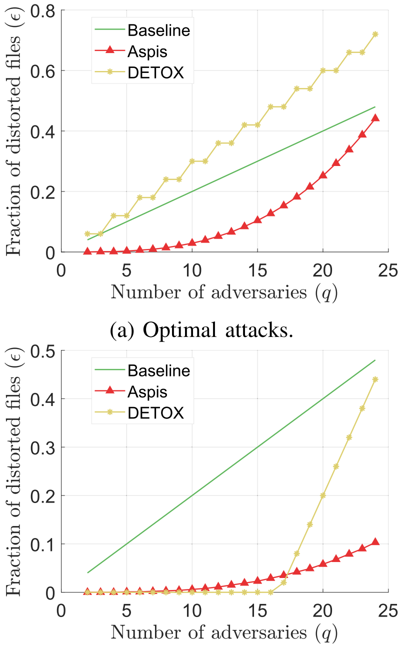
Remark 4: Using a $2-(v, 3, \lambda)$, i.e., a design with $k = 3$ (a typical value for the redundancy) to assign the files on a cluster with q Byzantines, the maximum number of files one can distort is $\lambda\binom{q}{2}/|\mathcal{B}|$ [33], where $|\mathcal{B}|$ is the total number of files; this is when each possible pair of Byzantines, among the $\binom{q}{2}$ possible ones appear together in a distinct block and distorts the corresponding file. In Aspis+, the focus is on weak attacks and determining the worst-case choice of adversaries that maximize the number of distorted files is beyond the scope of our work.

VI. Convergence Results and Experiments Under Setting-II

In this section, we operate under Setting-II (*cf.* Section III). By leveraging the work of Chen et al. [13] we demonstrate that our training algorithm converges to small enough neighborhood of the optimal point; the neighborhood size shrinks with the number of data samples.

We assume that the data samples are distributed i.i.d. from some unknown distribution μ. We are interested in finding $\mathbf{w}^*$ that minimizes $L(\mathbf{w}) = \mathbb{E}(l_1(\mathbf{w}))$ over the $\mathbf{w} \in \mathcal{W}$; here the expectation is over the distribution μ and the $l_i(\mathbf{w})$'s are distributed i.i.d as well. In general, since the distribution is unknown, $\mathbb{E}(l_1(\mathbf{w}))$ cannot be computed and we instead minimize the empirical loss function given by $\hat{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} l_i(\mathbf{w})$. We need the following additional assumptions.

In the discussion below, we say that a random vector $\mathbf{z}$ is sub-exponential with sub-exponential norm K if for every unit-vector v, $\mathbf{z}^\top v$ is a sub-exponential random variable with sub-exponential norm at most K, i.e., $\sup_{v:||v||\leq 1} Pr(|\mathbf{z}^\top v| > t) \leq \exp(-t/K)$ [43, Sec 2.7]. To keep notation simple, we reuse the letter C to denote different numerical constants in each use. This practice is common when working with classes of distributions such as sub-exponential.



(a) Optimal attacks.

(b) Weak attacks.

Fig. 3. Distortion fraction of optimal and weak attacks for $(K, r) = (50, 3)$ and comparison.

- The minimization of $\hat{L}(\mathbf{w})$ is performed by using Aspis or Aspis+ along with gradient descent (*cf.* Eq. (1)). This means that in Algorithm 2, the batch size $b = n$ for all iterations (*cf.* discussion after (2)). The robust estimator $\widehat{\text{med}}$ is the geometric median.
- The function $L(\mathbf{w})$ is $\beta-$strongly convex, and differentiable with respect to $\mathbf{w}$ with $\tilde{M}$-Lipschitz gradient. This means that for all $\mathbf{w}$ and $\mathbf{w}'$ we have
$$L(\mathbf{w}') \geq L(\mathbf{w}) + \nabla L(\mathbf{w})^T(\mathbf{w}' - \mathbf{w}) + \frac{\beta}{2}\|\mathbf{w} - \mathbf{w}'\|^2, \text{ and } \|\nabla L(\mathbf{w}) - \nabla L(\mathbf{w}')\| \leq \tilde{M}\|\mathbf{w} - \mathbf{w}'\|.$$
- The random vectors $\nabla l_i(\mathbf{w})$ for $i = 1, \ldots, n$ are sub-exponential with sub-exponential norm C. This assumption ensures that $\frac{1}{n}\sum_{i=1}^{n} \nabla l_i(\mathbf{w}^*)$ concentrates around its mean $\nabla L(\mathbf{w}^*) = 0$.
- Let $h_i(\mathbf{w}) = \nabla l_i(\mathbf{w}) - \nabla l_i(\mathbf{w}^*)$. For $i = 1, \ldots, n$, the random vectors $h_i(\mathbf{w})$ are sub-exponential with sub-exponential norm $C\|\mathbf{w} - \mathbf{w}^*\|$.
- For any $\delta \in (0, 1)$ there exists $\tilde{M}'$ (dependent on n and δ) that is non-increasing in n such that $\hat{L}(\mathbf{w})$ is $\tilde{M}'$-smooth with high probability, i.e,
$$P\left(\sup_{\mathbf{w},\mathbf{w}' \in \mathcal{W}:\mathbf{w} \neq \mathbf{w}'} \frac{\left\|\frac{1}{n}\sum_{i=1}^{n}(\nabla l_i(\mathbf{w}) - \nabla l_i(\mathbf{w}'))\right\|}{\|\mathbf{w} - \mathbf{w}'\|} \leq \tilde{M}'\right) \geq 1 - \frac{\delta}{3}.$$
Here $\mathcal{W}$ is the feasible parameter set.

We emphasize that such assumptions are similar to those of related literature that establishes convergence results. For example, ByGARS++ in [19] also assumes that loss is

c-strongly convex and gradient is locally Lipschitz; also, the adversaries merely use multiplicative noise. Zeno in [20] assumes: (i) the stochastic loss function (averaging n_r samples) is an unbiased estimator of the global loss, (ii) L-smoothness and μ-lower-bounded Taylor approximation of the loss function, and (iii) any correct gradient estimator has bounded variance.

For Aspis, Theorem 1 guarantees an upper bound on the fraction of corrupted gradients regardless of what attack is used. In particular, treating the majority logic and clique finding as a pre-processing step, we arrive at a set of f files, at most $c^{(q)}_{\max}$ (*cf.* Theorem 1) of which are "arbitrarily" corrupted. At this point, the PS applies the robust estimator $\widehat{\text{med}}$ - "geometric median" and uses it to perform the update step. We can leverage Theorem 5 of [13] to obtain the following result where d is the length of the parameter vector and for $p_i \in (0,1), i = 1,2$ the quantity $D(p_1||p_2) = p_1 \log_2(\frac{p_1}{p_2}) + (1-p_1)\log_2(\frac{1-p_1}{1-p_2})$.

Theorem 2: (adapted from [13]) Suppose that $\beta, \tilde{M}$ are all constants and $\log \tilde{M}' = \mathcal{O}(\log d)$. Assume that $\mathcal{W} \subset \{\mathbf{w} : \|\mathbf{w}-\mathbf{w}^*\| \leq \tilde{r}\sqrt{d}\}$ for positive $\tilde{r}$ such that $\log \tilde{r} = \mathcal{O}(d\log(n/f))$ and $2(1+\epsilon)c^{(q)}_{\max} \leq f$. Fix any $\alpha \in (c^{(q)}_{\max}/f, 1/2)$ and any $\delta > 0$ such that $\delta \leq \alpha - c^{(q)}_{\max}/f$ and $\log(1/\delta) = \mathcal{O}(d)$. There exist universal constants c_1, c_2 such that if

$$\frac{n}{f} \geq c_1 C_\alpha^2 d \log(n/f),$$

then with probability at least $1 - \exp(-fD(\alpha - \frac{c^{(q)}_{\max}}{f} \,||\, \delta))$, for all $t \geq 1$, the iterates of our algorithm with $\eta = \beta/(2\tilde{M}^2)$ satisfy

$$\|\mathbf{w}_t - \mathbf{w}^*\| \leq \left(\frac{1}{2} + \frac{1}{2}\sqrt{1 - \frac{\beta^2}{4\tilde{M}^2}}\right)^t \|\mathbf{w}_0 - \mathbf{w}^*\| + c_2\sqrt{\frac{df}{n}}. \tag{11}$$

An instance of a problem that satisfies the assumptions presented above is the linear regression problem. Formally, the data set consists of n vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\forall i, \mathbf{x}_i \in \mathbb{R}^d$. We construct the data matrix X of size $n \times d$ using these vectors as its rows. The n labels corresponding to the data points are computed as follows: $\mathbf{y} = X\mathbf{w}$, where $\mathbf{w}$ denotes the parameter set. For this problem, our loss function is the *least-squares* loss, i.e., we have $l_i(\mathbf{w}) = \frac{1}{2}(\mathbf{y}_i - \mathbf{x}_i^T\mathbf{w})^2$ for $i = 1, \ldots, n$ where $\mathbf{x}_i^T$ denotes the i^{th} row of X.

A. Numerical Experiments

We use the GD algorithm (1) with the initial randomly chosen parameter vector $\mathbf{w_0} \sim \mathcal{N}(\mathbf{0}_d, I_d)$. We partition the data matrix X row-wise into f submatrices $X_1, X_2, \ldots, X_f$, and correspondingly the label vector $\mathbf{y}$ into f sub-vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_f$, where f is the number of files of the distributed algorithm. A file $B_{t,i}$ consists of a pair $(X_i, \mathbf{y}_i)$. For each of its assigned files $B_{t_i} = (X_i, \mathbf{y}_i) \in \mathcal{N}^w(U_i)$, worker U_i either computes the honest partial gradient or a distorted value and returns it to the PS. Using the formulation of Section IV, the gradient in Eq. (3) for linear regression is the product $\mathbf{g}_{t,i} = X_i^T X_i \mathbf{w} - X_i^T \mathbf{y}_i$.

Metrics: For each scheme and value of q we run multiple Carlo simulations, and calculated the average least-square loss that each algorithm converges to across the Monte Carlo simulations. For each simulation we declare convergence if the final empirical loss is less than 0.1 We record the fraction of experiments that converged and the rate of convergence. In computing the average loss, the experiments that did not converged are not taken into account (for more details, please see Supplement Section XI-D).

1) Experiment Setup: In our experiments, we set $n = 50000$, $d = 100$ while our cluster consists of $K = 15$ workers. All replication-based schemes use $r = 3$. For Aspis+, we considered a $2-(15,3,1)$ design [33].

The geometric median is available as a Python library [44]. Initially, we tuned the learning rate for each scheme and each distortion method to decide the one to use for the Monte Carlo simulations; all learning rates 10^{-x}, $x = \{1, 2, \ldots, 6\}$ have been tested. Also, we fix the random seeds of our experiments to be the same across all schemes; this guarantees that the data matrix as well as the original model estimate $\mathbf{w_0}$ will be the same across all methods. At the beginning of the algorithm, all elements of X and $\mathbf{w}$ are generated randomly according to a $\mathcal{N}(0,1)$ distribution. For all runs, we chose to terminate the algorithm when the norm of gradient is less than 10^{-10} or the algorithm has reached a maximum number of 2000 iterations. Our code is available online.[4]

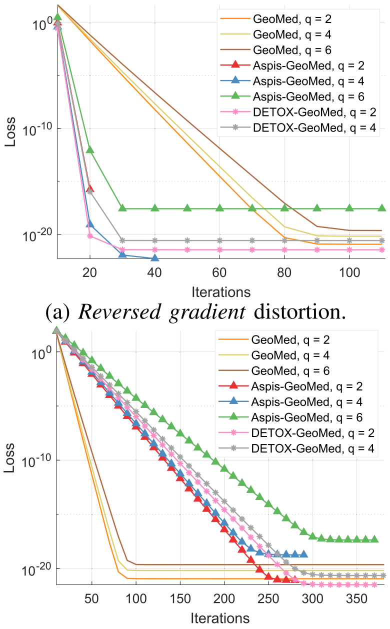
2) Results: The first set of experiments are for the strong attack ATT-2. The baseline scheme where geometric median is applied on all K gradients returned by the workers is referred to as *GeoMed* and it has no redundancy. DETOX, Aspis, and Aspis+ use geometric median as part of the robust aggregation. Under reversed gradient (see Figure 4a), it is clear that all schemes perform well and achieve similar loss for $q = 2, 4$ Byzantines. Nevertheless, baseline geometric median needed at least 100 iterations to converge while the redundancy-based schemes have a faster convergence rate. However, the situation is very different for $q = 6$ where Aspis converges within 30 iterations. In contrast, the DETOX loss diverged in all 100 simulations, while the convergence rate of the baseline scheme is much slower.

For the constant attack (see Fig. 4b) however, the relative performance of the baseline scheme and Aspis is reversed, i.e., the baseline scheme has a faster convergence rate as compared with Aspis. Moreover, Aspis and DETOX have roughly similar convergence rates for $q = 2, 4$. As before for $q = 6$, none of the DETOX simulations converged.

Our last set of results are with the ALIE attack and the results are reported in Figure 6. As the baseline geometric median simulations converged much slower than other schemes, they could not fit properly into the figure; it achieved final loss approximately equal to 2.07×10^{-28}, 9.71×10^{-28}, and 1.77×10^{-28} for $q = 2, 4$, and 6 respectively. On the other hand, Aspis converged to 1.69×10^{-23} in 30 iterations for $q = 6$. For this attack all schemes converged to very low loss values in all simulation runs. Nevertheless as is evident from Figure 6, both Aspis and DETOX converge to loss values less that 10^{-5} within 15 iterations.

Another experiment we performed compares our two proposed methods, Aspis and Aspis+. For both schemes,

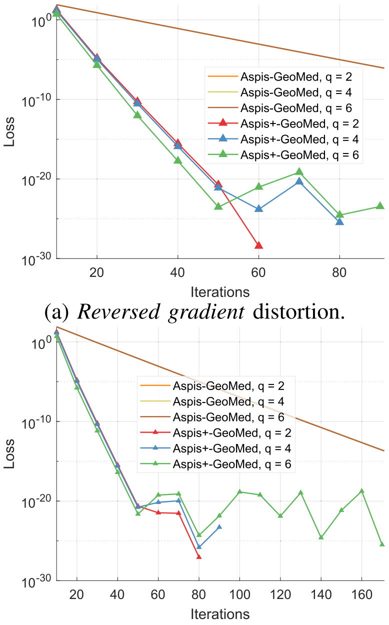
[4] https://github.com/kkonstantinidis/Aspis



(a) *Reversed gradient* distortion.

(b) *Constant* distortion.

Fig. 4. Linear regression least-squares loss, optimal attacks, ATT-2 (Aspis), geometric median defenses, $K = 15$.



(a) *Reversed gradient* distortion.

(b) *Constant* distortion.

Fig. 5. Linear regression least-squares loss, random Byzantines (ATT-3), geometric median defenses, $K = 15$.

we generate a new random Byzantine set A every $T_b = 50$ iterations (introduced as ATT-3 Section IV-C) while the detection window for Aspis+ is of length $T_d = 15$. For a comparable attack we use ATT-1 on Aspis (*cf.* Section IV-A), i.e., all adversaries distort all their assigned files. We compare the two schemes under reversed gradient attack in Figure 5a and under constant attack in Figure 5b. Both methods achieve low final loss in the order of 10^{-20} or lower; Aspis converged to lower losses of the order of 10^{-24} in approximately 280 iterations in all cases. Nevertheless, Aspis+ achieves a faster convergence rate which aligns with the fact that it's mostly suitable for weaker adversaries.

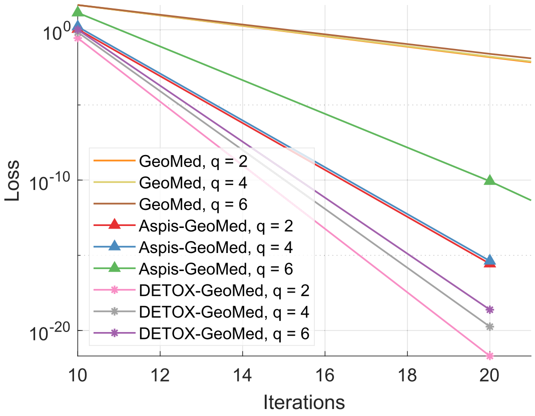


Fig. 6. Linear regression least-squares loss, *ALIE* distortion, optimal attacks, ATT-2 (Aspis), geometric median defenses, $K = 15$.

VII. Distortion Fraction Evaluation

The main motivation of our distortion fraction analysis is that our deep learning experiments (*cf.* Section VIII-B) and prior work [14] show that $\epsilon = c^{(q)}/f$ is a surrogate of the model's convergence with respect to accuracy. This comparison involves our work and state-of-the-art schemes under the best- and worst-case choice of the q adversaries in terms of the achievable value of ϵ. We also compare our work with *baseline* approaches that do not involve redundancy or majority voting and aggregation is applied directly to the K gradients returned by the workers ($f = K$, $c^{(q)}_{\max} = q$ and $\epsilon = q/K$).

For Aspis, we used the proposed attack ATT-2 from Section IV-B and the corresponding computation of $c^{(q),Aspis}$ of Theorem 1. *DETOX* in [16] employs a redundant assignment followed by majority voting and offers robustness guarantees which crucially rely on a "random choice" of the Byzantines. Our prior work [14] (ByzShield) has demonstrated the importance of a careful task assignment and observed that redundancy by itself is not sufficient to allow for Byzantine resilience. That work proposed an optimal choice of the q Byzantines that maximizes ϵ^{DETOX}, which we used in our current experiments. In short, DETOX splits the K workers into K/r groups. All workers within a group process the same subset of the batch, specifically containing br/K samples. This phase is followed by majority voting on a group-by-group basis. Reference [14] suggests choosing the Byzantines so that at least r' workers in each group are adversarial in order to distort the corresponding gradients. In this case, $c^{(q),DETOX} = \lfloor \frac{q}{r'} \rfloor$ and $\epsilon^{DETOX} = \lfloor \frac{q}{r'} \rfloor \times r/K$. We also compare with the distortion fraction incurred by ByzShield [14] under a worst-case scenario. For this scheme, there is no known optimal attack, and we performed an exhaustive combinatorial search to find the q adversaries that maximize $\epsilon^{ByzShield}$ among all possible options; we follow the same process here to simulate ByzShield's distortion fraction computation while utilizing the scheme of that work based on *mutually orthogonal Latin squares*. The reader can refer to Figure 3a and Supplement Tables III, IV, and V for our results. Aspis achieves major reductions in ϵ; for instance, $\epsilon^{Aspis,ATT-2}$ is reduced by up to 99% compared to both $\epsilon^{Baseline}$ and ϵ^{DETOX} in Figure 3a.

Next, we consider the *weak* attack, ATT-1. For our scheme, we will make an arbitrary choice of q adversaries which carry out the method introduced in Section V-A.1, i.e., they will distort all files, and a successful detection is possible. As discussed in Section V-A.1, the fraction of corrupted gradients is $\epsilon^{Aspis,ATT-1} = \binom{q}{r}/\binom{K}{r}$. For DETOX, a simple benign attack is used. To that end, let the K/r files be $B_{t,0}, B_{t,1}, \ldots, B_{t,K/r-1}$. Initialize $A = \emptyset$ and choose the q Byzantines as follows: for $i = 0, 1, \ldots, q-1$, among the remaining workers in $\{U_1, U_2, \ldots, U_K\} - A$ add a worker from the group $B_{t,i \bmod K/r}$ to the adversarial set A. Then,

$$c^{(q),DETOX} = \begin{cases} q - \dfrac{K}{r}(r'-1) & \text{if } q > \dfrac{K}{r}(r'-1), \\ 0 & \text{otherwise.} \end{cases}$$

The results of this scenario are in Figure 3.

VIII. Large-Scale Deep Learning Experiments

All these experiments are performed under Setting-I, i.e., no assumptions are made about the dataset or the loss function. Accordingly, the evaluation here is in terms of the distortion fraction (see Section VII) and numerical experiments (described below). For the experiments, we used the mini-batch SGD (see (2)) and the robust estimator (see Algorithm 2) is the coordinate-wise median.

A. Experiment Setup

We have evaluated the performance of our methods and competing techniques in classification tasks on Amazon EC2 clusters. The project is written in PyTorch [1] and uses the MPICH library for communication between the different nodes. We worked with the CIFAR-10 data set [32] using the ResNet-18 [45] model. We used clusters of $K = 15$, 21, 25 workers, redundancy $r = 3$, and simulated values of $q = 2, 4, 6, 7, 9$ during training. Detailed information about the implementation can be found in Supplement Section XI-A.

Competing methods: We compare Aspis against the baseline implementations of median-of-means [46], Bulyan [24], and Multi-Krum [12]. If $c_{\max}^{(q)}$ is the number of adversarial computations, then Bulyan requires at least $4c_{\max}^{(q)} + 3$ total number of computations while the same number for Multi-Krum is $2c_{\max}^{(q)} + 3$. These constraints make these methods inapplicable for larger values of q for which our methods are robust. The second class of comparisons is with methods that use redundancy, specifically DETOX [16]. For the baseline scheme we compare with median-based techniques since they originate from robust statistics and are the basis for many aggregators. Multi-Krum combines the intuitions of majority-based and squared-distance-based methods. Draco [17] is a closely related method that uses redundancy. However we do not compare with it since it is very limited in the number of Byzantines that it is resilient to.

Note that for a baseline scheme, all choices of A are equivalent in terms of the value of ϵ. In our comparisons between Aspis and DETOX we will consider two attack scenarios concerning the choice of the adversaries. For the *optimal* attack on DETOX, we will use the method proposed in [14] and compare with the attack introduced in Section V-A.2. For the *weak* one, we will choose the adversaries such that they incur the minimum value of ϵ in DETOX for given q and compare its performance with the scenario of Section V-A.1. All schemes compared with Aspis+ consider random sets of Byzantines, and for Aspis+, we will use the attack ATT-3.

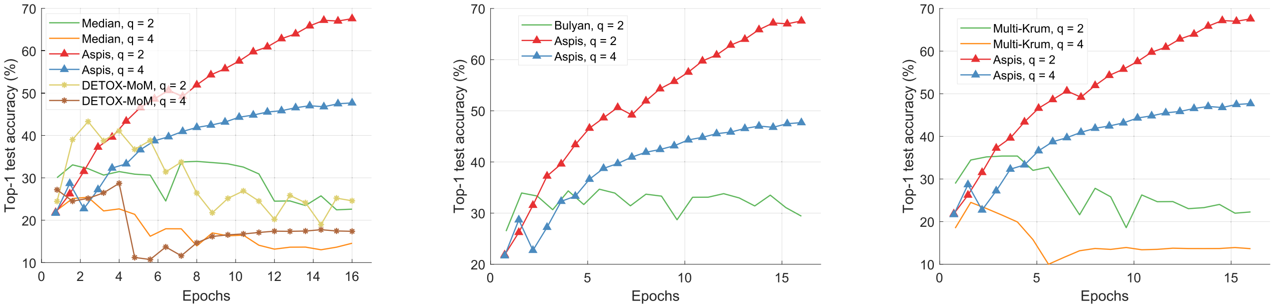
B. Aspis Experimental Results

1) Comparison Under Optimal Attacks: We compare the different defense algorithms under optimal attack scenarios using ATT-2 for Aspis. Figure 7a compares our scheme Aspis with the baseline implementation of coordinate-wise median ($\epsilon = 0.133, 0.267$ for $q = 2, 4$, respectively) and DETOX with median-of-means ($\epsilon = 0.2, 0.4$ for $q = 2, 4$, respectively) under the ALIE attack. Aspis converges faster and achieves at least a 35% average accuracy boost (at the end of the training) for both values of q ($\epsilon^{Aspis} = 0.004, 0.062$ for $q = 2, 4$, respectively).[5] In Figures 7b and 7c, we observe similar trends in our experiments with Bulyan and Multi-Krum, where Aspis significantly outperforms these techniques. For the current setup, Bulyan is not applicable for $q = 4$ since $K = 15 < 4c_{\max}^{(q)} + 3 = 4q + 3 = 19$. Also, neither Bulyan nor Multi-Krum can be paired with DETOX for $q \geq 4$ since the inequalities $f \geq 4c_{\max}^{(q)} + 3$ and $f \geq 2c_{\max}^{(q)} + 3$, where $f = f_{\text{DETOX}} = K/r$, cannot be satisfied; for the specific case of Bulyan even $q = 2, 3$ would not be supported by DETOX. Please refer to Section VIII-A and Section VII for more details on these requirements. Also, note that the accuracy of most competing methods fluctuates more than in the results presented in the corresponding papers [16] and [23]. This is expected as we consider stronger attacks than those papers, i.e., optimal deterministic attacks on DETOX and, in general, up to 27% adversarial workers in the cluster. Also, we have done multiple experiments with different random seeds to demonstrate the stability and superiority of our accuracy results compared to other methods (against median-based defenses in Supplement Figure 15, Bulyan in Supplement Figure 16 and Multi-Krum in Supplement Figure 17); we point the reader to Supplement Section 17 for this analysis. This analysis is clearly missing from most prior work, including that of ALIE [23] and their presented results are only a snapshot of a single experiment. The results for the reversed gradient attack are shown in Figures 8a, 8b, and 8c. Given that this is a weaker attack [14], [16] all schemes, including the baseline methods, are expected to perform well; indeed, in most cases, the model converges to approximately 80% accuracy. However, DETOX fails to converge to high accuracy for $q = 4$ as in the case of ALIE; one explanation is that $\epsilon^{DETOX} = 0.4$ for $q = 4$. Under the Fall of Empires (FoE) distortion (*cf.* Figure 9) our method still enjoys an accuracy advantage over the baseline and DETOX schemes which becomes more important as the number of Byzantines in the cluster increases.

We have also performed experiments on larger clusters ($K = 21$ workers) as well. The results for the ALIE distortion with the ATT-2 attack can be found in Figure 12. They exhibit similar behavior as in the case of $K = 15$.

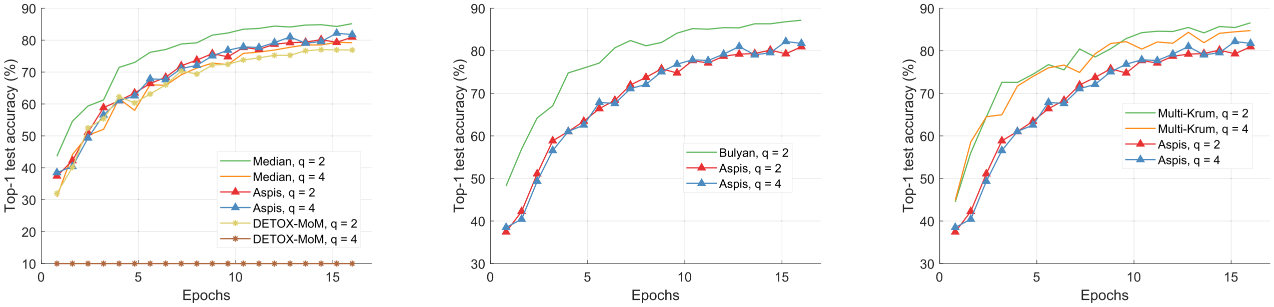
2) Comparison Under Weak Attacks: For baseline schemes, the discussion of weak versus optimal choice of the adversaries is not very relevant as any choice of the q Byzantines can overall distort exactly q out of the K gradients. Hence, for weak scenarios, we chose to compare mostly with DETOX while using ATT-1 on Aspis. The accuracy is reported in Figures 10 and 11, according to which Aspis shows an improvement under attacks on the more challenging end of the

[5] Please refer to Supplement Table III(a) for the values of the distortion fraction ϵ each scheme incurs.



(a) Median-based defenses. (b) *Bulyan*-based defenses. (c) *Multi-Krum*-based defenses.

Fig. 7. *ALIE* distortion under optimal attack scenarios, ATT-2 for Aspis, CIFAR-10, $K = 15$.



(a) Median-based defenses. (b) *Bulyan*-based defenses. (c) *Multi-Krum*-based defenses.

Fig. 8. *Reversed gradient* distortion under optimal attack scenarios, ATT-2 for Aspis, CIFAR-10, $K = 15$.

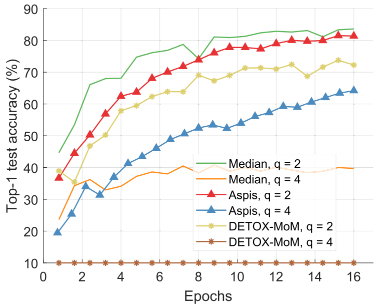


Fig. 9. *FoE* distortion, optimal attacks, ATT-2 (Aspis) and median-based defenses (CIFAR-10), $K = 15$.

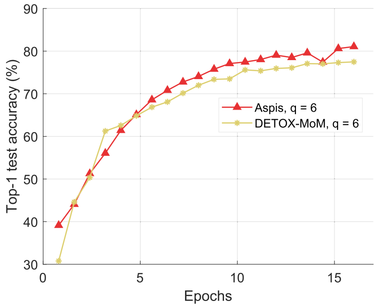


Fig. 10. *Reversed gradient* distortion, weak attacks, ATT-1 (Aspis) and median-based defenses (CIFAR-10), $K = 15$.

spectrum (ALIE). According to Supplement Table III(b), Aspis enjoys a fraction $\epsilon^{Aspis} = 0.044$ while $\epsilon^{Baseline} = 0.4$ and $\epsilon^{DETOX} = 0.2$ for $q = 6$.

C. Aspis+ Experimental Results

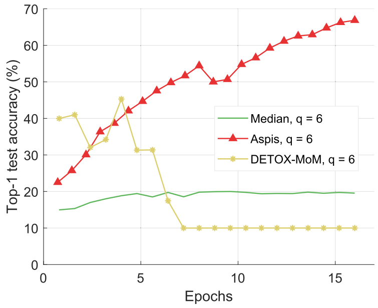


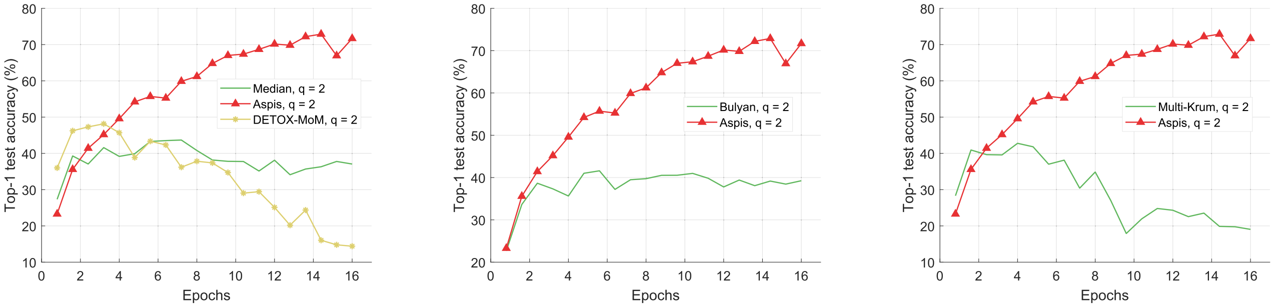
Fig. 11. *ALIE* distortion, weak attacks, ATT-1 (Aspis) and median-based defenses (CIFAR-10), $K = 15$.

For Aspis+, we considered the attack ATT-3 discussed in Section IV-C. We tested clusters of $K = 15$ with $q = 2, 4$ and $K = 25$ workers among which $q = 7, 9$ are Byzantine. In the former case, a $2 - (15, 3, 1)$ design [33] with $f = 35$ blocks (files) was used for the placement, while in the latter case, we used a $2 - (25, 3, 1)$ design [33] with $f = 100$ blocks (files). A new random Byzantine set A is generated every $T_b = 50$ iterations while the detection window is of length $T_d = 15$.

The results for $K = 15$ are in Figure 13. We tested against the ALIE distortion, and all compared methods use median-based defenses to filter the gradients. Aspis+ demonstrates an advantage of at least 15% compared with other algorithms (*cf.* $q = 2$). For $K = 25$, we tried a weaker distortion than ALIE, i.e., the constant attack paired with *signSGD*-based defenses [26]. In signSGD, the PS will output the majority of the gradients' signs for each dimension. Following the advice of [16], we pair this defense with the stronger constant attack as sign flips (*e.g.,* reversed gradient) are unlikely to affect the gradient's distribution. Aspis+ with median still enjoys an accuracy improvement of at least 20% for $q = 7$ and a larger one for $q = 9$. The results are in Figure 14; in this figure, the DETOX accuracy is an average of two experiments using two different random seeds.

IX. Conclusion and Future Work

In this work, we have presented Aspis and Aspis+, two Byzantine-resilient distributed schemes that use redundancy



(a) Median-based defenses. (b) *Bulyan*-based defenses. (c) *Multi-Krum*-based defenses.

Fig. 12. *ALIE* distortion under optimal attack scenarios, ATT-2 for Aspis, CIFAR-10, $K = 21$.

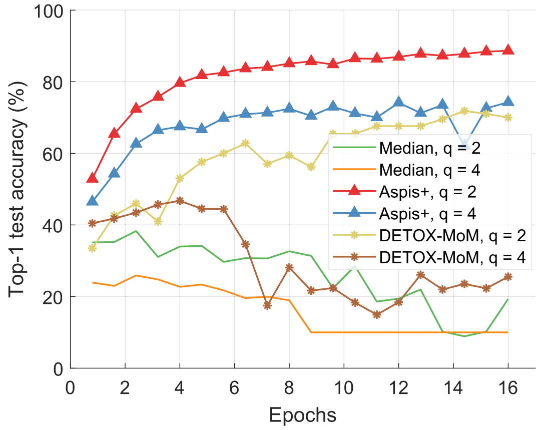


Fig. 13. *ALIE* distortion and random Byzantines, $K = 15$ (median-based defenses). ATT-3 used on Aspis+.

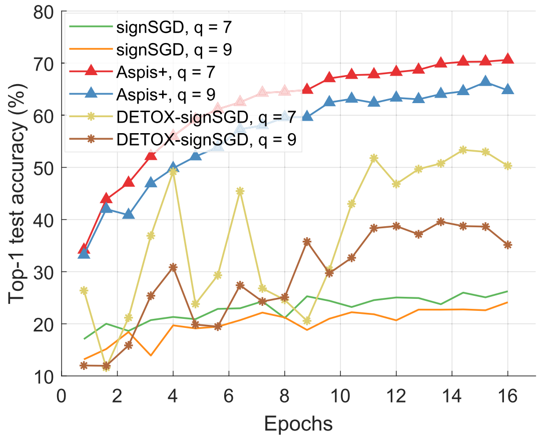


Fig. 14. *Constant* distortion and random Byzantines, $K = 25$ (*signSGD*-based defenses). ATT-3 used on Aspis+.

and robust aggregation in novel ways to detect failures of the workers. Our theoretical analysis and numerical experiments clearly indicate their superior performance compared to state-of-the-art. Our experiments show that these methods require increased computation and communication time as compared to prior work, e.g., note that each worker has to transmit l gradients instead of 1 in related work [16], [17] (see Supplement Section XI-A4 for details). We emphasize, however, that our schemes converge to high accuracy in our experiments, while other methods remain at much lower accuracy values regardless of how long the algorithm runs for.

Our experiments involve clusters of up to 25 workers. As we scale Aspis to more workers, the total number of files and the computation load l of each worker will also scale; this increases the memory needed to store the gradients during aggregation. For complex neural networks, the memory to store the model and the intermediate gradient computations is by far the most memory-consuming aspect of the algorithm. For these reasons, Aspis is mostly suitable for training large data sets using fairly compact models that do not require too much memory. Aspis+, on the other hand, is a good fit for clusters that suffer from non-adversarial failures that can lead to inaccurate gradients. Finally, utilizing GPUs and investigating algorithmic communication-related improvements are worth exploring to reduce the time overhead.

Appendix

A. Proof of Theorem 1

For a given file F, let $A' \subseteq A$ with $|A'| \geq r'$ be the set of "active adversaries" in it, i.e., $A' \subseteq F$ consists of Byzantines that collude to create a majority that distorts the gradient on it. In this case, the remaining workers in F belong to $\cap_{i \in A'} D_i$, where we note that $|\cap_{i \in A'} D_i| \leq q$. Let $X_j, j = r', r'+1, \ldots, r$ denote the subset of files where the set of active adversaries is of size j; note that X_j depends on the disagreement sets $D_i, i = 1, 2, \ldots, q$. Formally,

$$X_j = \{F : \exists A' \subseteq A \cap F, |A'| = j, \\ \text{and } \forall\ U_j \in F \setminus A', U_j \in \cap_{i \in A'} D_i\}. \qquad (12)$$

Then, for a given choice of disagreement sets, the number of files that can be corrupted is given by $| \cup_{j=r'}^{r} X_j|$. We obtain an upper bound on the maximum number of corrupted files by maximizing this quantity with respect to the choice of $D_i, i = 1, 2, \ldots, q$, i.e.,

$$c_{\max}^{(q)} = \max_{D_i, |D_i| \leq q, i=1,2,\ldots,q} | \cup_{j=r'}^{r} X_j| \qquad (13)$$

where the maximization is over the choice of the disagreement sets $D_1, D_2, \ldots, D_q$. With X_j given in (12), assuming $q \geq r'$, the number of distorted files is upper bounded by

$$| \cup_{j=r'}^{r} X_j| \leq \sum_{j=r'}^{r} |X_j| \text{ (by the union bound).} \qquad (14)$$

For that, recall that $r' = (r+1)/2$ and that an adversarial majority of at least r' distorted computations for a file is needed to corrupt that particular file. Note that X_j consists of those files where the active adversaries A' are of size j; these can be chosen in $\binom{q}{j}$ ways. The remaining workers in the file

TABLE II
Main Notation of the Paper

Symbol	Meaning
K	number of workers
q	number of adversaries
r	redundancy (number of workers each file is assigned to)
b	batch size
B_t	samples of batch of t^{th} iteration
f	number of files (alternatively called *groups* or *tasks*)
U_j	j^{th} worker
l	computation load (number of files per worker)
$\mathcal{N}^w(U_j)$	set of files of worker U_j
$\mathcal{N}^f(B_{t,i})$	set of workers assigned to file $B_{t,i}$
$\mathbf{g}_{t,i}$	true gradient of file $B_{t,i}$ with respect to $\mathbf{w}$
$\hat{\mathbf{g}}_{t,i}^{(j)}$	returned gradient of U_j for file $B_{t,i}$ with respect to $\mathbf{w}$
$\mathbf{m}_i$	majority gradient for file $B_{t,i}$
$\mathcal{U}$	worker set $\{U_1, U_2, ..., U_K\}$
$\mathbf{G}_{task}$	graph used to encode the task assignments to workers
$\mathbf{G}_t$	graph indicating the agreements of pairs of workers in all of their common gradient tasks in t^{th} iteration
A	set of adversaries
$M_{\mathbf{G}}$	maximum clique in $\mathbf{G}$
$c^{(q)}$	number of distorted gradients after detection and aggregation
$c_{\max}^{(q)}$	maximum number of distorted gradients after detection and aggregation (worst-case)
D_i	disagreement set (of workers) for i^{th} adversary
r'	$(r+1)/2$, i.e., minimum number of distorted copies needed to corrupt majority vote for a file
ϵ	$c^{(q)}/f$, i.e., fraction of distorted gradients after detection and aggregation
X_j	subset of files where the set of active adversaries is of size j; for linear regression this is the data matrix corresponding to the i^{th} file
X	data matrix of linear regression
n	number of points of linear regression
d	dimensionality of linear regression model

belong to $\cap_{i \in A'} D_i$ where $|\cap_{i \in A'} D_i| \leq q$. Thus, the remaining workers can be chosen in at most $\binom{q}{r-j}$ ways. It follows that

$$|X_j| \leq \binom{q}{j}\binom{q}{r-j}. \tag{15}$$

Therefore,

$$\begin{aligned} c_{\max}^{(q)} \leq &\binom{q}{r'}\binom{q}{r-r'} + \binom{q}{r'+1}\binom{q}{r-(r'+1)} \\ &+ \cdots \\ &+ \binom{q}{r-1}\binom{q}{r-(r-1)} + \binom{q}{r} && (16) \\ = &\sum_{i=r'}^{q}\binom{q}{i}\binom{q}{r-i} && (17) \\ = &\sum_{i=0}^{q}\binom{q}{i}\binom{q}{r-i} - \sum_{i=0}^{r'-1}\binom{q}{i}\binom{q}{r-i} && (18) \\ = &\frac{1}{2}\binom{2q}{r}. && (19) \end{aligned}$$

Eq. (17) follows from the convention that $\binom{n}{k} = 0$ when $k > n$ or $k < 0$. Eq. (19) follows from Eq. (18) using the following observations

- $\sum_{i=0}^{q}\binom{q}{i}\binom{q}{r-i} = \sum_{i=0}^{r}\binom{q}{i}\binom{q}{r-i} = \binom{2q}{r}$ in which the first equality is straightforward to show by taking all possible cases: $q < r$, $q = r$ and $q > r$.
- By symmetry, $\sum_{i=0}^{r'-1}\binom{q}{i}\binom{q}{r-i} = \sum_{i=r'}^{q}\binom{q}{i}\binom{q}{r-i} = \frac{1}{2}\binom{2q}{r}$.

The upper bound in Eq. (16) is met with equality when all adversaries choose the same disagreement set, which is a q-sized subset of the honest workers, i.e., $D_i = D \subset H$ for $i = 1, \ldots, q$. In this case, it can be seen that the sets $X_j, j = r', \ldots, r$ are disjoint so that (14) is met with equality. Moreover, (15) is also an equality. This finally implies that (16) is also an equality, i.e., this choice of disagreement sets saturates the upper bound.

It can also be seen that in this case, the adversarial strategy yields a graph $\mathbf{G}$ with multiple maximum cliques. To see this, we note that the adversaries in A agree with all the computed gradients in $H \setminus D$. Thus, they form a clique of $M_{\mathbf{G}}^{(1)}$ of size $K - q$ in $\mathbf{G}$. Furthermore, the honest workers in H form another clique $M_{\mathbf{G}}^{(2)}$, which is also of size $K-q$. Thus, the detection algorithm cannot select one over the other and the adversaries will evade detection; and the fallback robust aggregation strategy will apply.

References

[1] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2019, pp. 8024–8035.

[2] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. OSDI*, Nov. 2016, pp. 265–283.

[3] T. Chen et al., "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015, *arXiv:1512.01274*.

[4] F. Seide and A. Agarwal, "CNTK: Microsoft's open-source deep-learning toolkit," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, p. 2135.

[5] Y. Kim et al., "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *Proc. ACM/IEEE 41st Int. Symp. Comput. Archit. (ISCA)*, Jun. 2014, pp. 361–372.

[6] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1211–1220.

[7] N. Gupta and N. H. Vaidya, "Byzantine fault-tolerant parallelized stochastic gradient descent for linear regression," in *Proc. 57th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2019, pp. 415–420.

[8] G. Damaskinos, E. M. E. Mhamdi, R. Guerraoui, A. H. A. Guirguis, and S. L. A. Rouault, "Aggregathor: Byzantine machine learning via robust gradient aggregation," in *Proc. Conf. Syst. Mach. Learn. (SysML)*, Mar. 2019, p. 19.

[9] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Defending against saddle point attack in Byzantine-robust distributed learning," in *Proc. 36th Int. Conf. Mach. Learn.*, Jun. 2019, pp. 7074–7084.

[10] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. ICML*, Jul. 2018, pp. 5650–5659.

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

[12] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst. Annu. Conf. Neural Inf. Process. Syst.*, Dec. 2017, pp. 119–129.

[13] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, Dec. 2017, Art. no. 44, doi: 10.1145/3154503.

[14] K. Konstantinidis and A. Ramamoorthy, "ByzShield: An efficient and robust system for distributed training," in *Proc. MLSys*, Apr. 2021, pp. 812–828.

[15] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. Avestimehr, “Lagrange coded computing: Optimal design for resiliency, security and privacy,” 2018, *arXiv:1806.00939*.

[16] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, “DETOX: A redundancy-based framework for faster and more robust gradient aggregation,” in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2019, pp. 10320–10330.

[17] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, “DRACO: Byzantine-resilient distributed training via redundant gradients,” in *Proc. 35th Int. Conf. Mach. Learn.*, Jul. 2018, pp. 903–912.

[18] D. Data, L. Song, and S. Diggavi, “Data encoding for Byzantine-resilient distributed gradient descent,” in *Proc. 56th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2018, pp. 863–870.

[19] J. Regatti, H. Chen, and A. Gupta, “ByGARS: Byzantine SGD with arbitrary number of attackers,” 2020, *arXiv:2006.13421*.

[20] C. Xie, S. Koyejo, and I. Gupta, “Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance,” in *Proc. 36th Int. Conf. Mach. Learn.*, Jun. 2019, pp. 6893–6901.

[21] D. Alistarh, Z. Allen-Zhu, and J. Li, “Byzantine stochastic gradient descent,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran, Dec. 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/a07c2f3b3b907aaf8436a26c6d77f0a2-Paper.pdf

[22] K. Konstantinidis and A. Ramamoorthy, “Aspis: Robust detection for distributed learning,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 2058–2063.

[23] G. Baruch, M. Baruch, and Y. Goldberg, “A little is enough: Circumventing defenses for distributed learning,” in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2019, pp. 8635–8645.

[24] E. M. El Mhamdi, R. Guerraoui, and S. Rouault, “The hidden vulnerability of distributed learning in Byzantium,” in *Proc. 35th Int. Conf. Mach. Learn.*, Jul. 2018, pp. 3521–3530.

[25] S. Shen, S. Tople, and P. Saxena, “AUROR: Defending against poisoning attacks in collaborative deep learning systems,” in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, Dec. 2016, pp. 508–519.

[26] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, “SignSGD with majority vote is communication efficient and fault tolerant,” 2018, *arXiv:1810.05291*.

[27] J. So, B. Güler, and A. S. Avestimehr, “Byzantine-resilient secure federated learning,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2021.

[28] R. Jin, Y. Huang, X. He, H. Dai, and T. Wu, “Stochastic-sign SGD for federated learning with theoretical guarantees,” 2020, *arXiv:2002.10940*.

[29] N. Raviv, R. Tandon, A. Dimakis, and I. Tamo, “Gradient coding from cyclic MDS codes and expander graphs,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jul. 2018, pp. 4302–4310.

[30] R. Tandon, Q. Lei, A. Dimakis, and N. Karampatziakis, “Gradient coding: Avoiding stragglers in distributed learning,” in *Proc. 34th Int. Conf. Mach. Learn.*, Aug. 2017, pp. 3368–3376.

[31] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, Jan. 2018.

[32] A. Krizhevsky, “Learning multiple layers of features from tiny images,” M.S. thesis, Dept. CS, Univ. Toronto, Toronto, ON, Canada, 2009.

[33] D. R. Stinson, *Combinatorial Designs: Constructions and Analysis*. New York, NY, USA: Springer, 2004.

[34] C. Xie, O. Koyejo, and I. Gupta, “Fall of empires: Breaking Byzantine-tolerant SGD by inner product manipulation,” in *Proc. 35th Conf. Uncertainty Artif. Intell.*, Jul. 2019, pp. 6893–6901.

[35] R. M. Karp, *Reducibility Among Combinatorial Problems*. Boston, MA, USA: Springer, 1972.

[36] F. Cazals and C. Karande, “A note on the problem of reporting maximal cliques,” *Theor. Comput. Sci.*, vol. 407, nos. 1–3, pp. 564–568, Nov. 2008.

[37] E. Tomita, A. Tanaka, and H. Takahashi, “The worst-case time complexity for generating all maximal cliques and computational experiments,” *Theor. Comput. Sci.*, vol. 363, no. 1, pp. 28–42, Oct. 2006.

[38] J. M. Robson, “Algorithms for maximum independent sets,” *J. Algorithms*, vol. 7, no. 3, pp. 425–440, Sep. 1986.

[39] R. E. Tarjan and A. E. Trojanowski, “Finding a maximum independent set,” *SIAM J. Comput.*, vol. 6, no. 3, pp. 537–546, Sep. 1977.

[40] D. Masters and C. Luschi, “Revisiting small batch training for deep neural networks,” 2018, *arXiv:1804.07612*.

[41] Y. You, I. Gitman, and B. Ginsburg, “Large batch training of convolutional networks,” 2017, *arXiv:1708.03888*.

[42] J. Geiping, M. Goldblum, P. E. Pope, M. Moeller, and T. Goldstein, “Stochastic training is not necessary for generalization,” 2021, *arXiv:2109.14119*.

[43] R. Vershynin, *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge, U.K.: Cambridge Univ. Press, 2018.

[44] K. Pillutla, S. M. Kakade, and Z. Harchaoui, “Robust aggregation for federated learning,” *IEEE Trans. Signal Process.*, vol. 70, pp. 1142–1154, 2022.

[45] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[46] S. Minsker, “Geometric median and robust estimation in Banach spaces,” *Bernoulli*, vol. 21, no. 4, pp. 2308–2335, Nov. 2015.

[47] R. S. Boyer and J. S. Moore, *MJRTY—A Fast Majority Vote Algorithm*. Dordrecht, The Netherlands: Springer, 1991.

[48] (Sep. 2022). *NVIDIA CUDA Toolkit*. [Online]. Available: https://developer.nvidia.com/cuda-toolkit

[49] (Aug. 2022). *Repository of ByzShield Implementation*. [Online]. Available: https://github.com/kkonstantinidis/ByzShield

[50] A. Hagberg, P. Swart, and D. S. Chult, “Exploring network structure, dynamics, and function using NetworkX,” in *Proc. 7th Python Sci. Conf.*, Jan. 2008, pp. 11–15.

Konstantinos Konstantinidis received the Diploma degree in electrical and computer engineering from the Technical University of Crete, Greece, in 2016, and the Ph.D. degree in electrical and computer engineering from Iowa State University, Ames, IA, USA, in 2022, under the supervision of Prof. Aditya Ramamoorthy. His thesis, while being a bachelor’s student, focused on blind synchronization and detection of binary frequency-shift keying (BFSK) signals. His Ph.D. thesis and current research interests include communication load reduction in distributed systems, network coding, and distributed computing for machine learning applications.

Namrata Vaswani (Fellow, IEEE) received the B.Tech. degree in electrical engineering from IIT-Delhi, India, in 1999, and the Ph.D. degree in electrical engineering from the University of Maryland, College Park, MD, USA, in 2004. Since Fall 2005, she has been with Iowa State University, Ames, IA, USA, where she is currently the Anderlik Professor of electrical and computer engineering. Her research interests include data science, with a particular focus on statistical machine learning and signal processing. She was a recipient of the Iowa State Early Career Engineering Faculty Research Award in 2014, the Iowa State University Mid-Career Achievement in Research Award in 2019, and the University of Maryland’s ECE Distinguished Alumni Award in 2019. She also received the 2014 IEEE Signal Processing Society Best Paper Award for her 2010 IEEE TRANSACTIONS ON SIGNAL PROCESSING paper coauthored with her student Wei Lu. She has served as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY and the IEEE TRANSACTIONS ON SIGNAL PROCESSING, as a Lead Guest-Editor for a 2018 PROCEEDINGS OF THE IEEE Special Issue (Rethinking PCA for modern datasets), and as an Area Editor for Special Issues for the *IEEE Signal Processing Magazine*.

Aditya Ramamoorthy (Senior Member, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Delhi and the M.S. and Ph.D. degrees from the University of California at Los Angeles. He is currently the Northrop Grumman Professor of electrical and computer engineering and (by courtesy) of mathematics with Iowa State University. His research interests include classical/quantum information theory and coding techniques with applications to distributed computation, content distribution networks, and machine learning. He was a recipient of the 2020 Mid-Career Achievement in Research Award, the 2019 Boast-Nilsson Educational Impact Award, and the 2012 Early Career Engineering Faculty Research Award from Iowa State University, the 2012 NSF CAREER Award, and the Harpole-Pentair Professorship in 2009 and 2010. He currently serves as an editor for the IEEE TRANSACTIONS ON INFORMATION THEORY (previous term from 2016 – 2019) and was an editor for the IEEE TRANSACTIONS ON COMMUNICATIONS from 2011 – 2015.