Aspis: Robust Detection for Distributed Learning

Konstantinos Konstantinidis and Aditya Ramamoorthy
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50010
Email: {kostas, adityar}@iastate.edu

2022 IEEE International Symposium on Information Theory (ISIT) | 978-1-6654-2159-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISIT50566.2022.9834813

***Abstract*—State-of-the-art machine learning models are routinely trained on large-scale distributed clusters. Crucially, such systems can be compromised when some of the computing devices exhibit abnormal (*Byzantine*) behavior and return arbitrary results to the *parameter server* (PS). This behavior may be attributed to a plethora of reasons, including system failures and orchestrated attacks. Existing work suggests *robust aggregation* and/or computational *redundancy* to alleviate the effect of distorted gradients. However, most of these schemes are ineffective when an adversary knows the task assignment and can choose the attacked workers judiciously to induce maximal damage. Our proposed method *Aspis* assigns gradient computations to workers using a subset-based assignment which allows for multiple consistency checks on the behavior of a worker. Examination of the calculated gradients and clique-finding in an appropriately constructed graph by the PS allows for efficient detection and exclusion of adversaries from the training. We prove the Byzantine resilience guarantees of Aspis under weak and strong attacks and extensively evaluate the system on various training scenarios and demonstrate an improvement of about 30% in accuracy compared to many state-of-the-art approaches on the CIFAR-10 dataset as well as reduction of the fraction of corrupted gradients ranging from 16% to 99%.**

I. Introduction

The increased sizes of datasets and associated model complexities have established distributed training setups as the de facto method for training models at scale. A typical setup consists of one *parameter server* (PS) and multiple workers. The PS coordinates the protocol by communicating parameters and maintaining the model. The workers compute gradients of the loss function with respect to the optimization parameters and transmit them to the PS. The PS then updates the model. This is an iterative process repeated until convergence.

Despite their speedup benefits, such distributed settings are prone to so-called *Byzantine* failures, i.e., when a set of workers return malicious or erroneous computations. This can happen on purpose due to adversarial attacks or inadvertently due to hardware or software failures. For example, [1] showed that bit-flips in commodity DRAM can happen merely through frequent data access of the same address. Reference [2] exposes the vulnerability of neural networks to such failures and identifies weight parameters that could maximize accuracy degradation. As a result, the distorted gradients can derail the optimization and lead to low test accuracy. Devising training algorithms that are resilient to such failures and which can efficiently *aggregate* the gradients has inspired a series of works [3], [4], [5], [6].

Prior work on *robust aggregation* [5], [6], [7], [8], [9], [10] provides robustness guarantees up to a constant fraction of the nodes being adversarial. However, this fraction is usually very small, and the guarantees are limited (*e.g.,* only guaranteeing that the output of the aggregator has a positive inner product with the true gradient [6], [11]). Also, they require significant asymptotic complexity [10] and strict convexity assumptions that need to be adjusted for each individual training algorithm. *Redundancy*-based schemes assign each gradient task to more than one node [12], [13], [14], [15], [16]. Existing techniques are sometimes combined with robust aggregation [14]. Fundamentally, these methods require a higher computation load per worker, but they come with stronger guarantees of correcting the erroneous gradients. Most schemes in this category can be made to fail by a powerful, *omniscient* adversary that can mount judicious attacks [12]. Another line of work focuses on *ranking* and/or *detection* of the adversaries [15], [17], [18]; 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. Their theoretical guarantees require strict assumptions on the smoothness of the loss and the authors have not used or constructed worst-case attacks to evaluate the methods in adversarial settings.

A. Contributions

Our scheme Aspis uses a combination of *redundancy* and *robust aggregation*. Unlike previous methods, the redundant subset-based assignment for gradient computations is judiciously chosen such that the PS can perform global *consistency checks* on the workers by examining the returned gradients. Clique-finding in appropriate graphs is used by the PS for *detection* to exclude adversaries from the training.

Under weak attacks where the Byzantines act independently, they will always be detected by the proposed novel clique-based algorithm. Aspis is resilient to stronger attacks (*optimal* collusion) than those considered in prior work. Instead of simulating a random set of adversaries [14], [15], we have crafted a non-trivial attack such that the adversaries can evade our detection and corrupt more gradients.

We provide theoretical guarantees for both weak and strong attacks on the fraction of corrupted gradients for Aspis. Comparisons with other methods indicate reductions in the fraction of corrupted gradients ranging from 16% to 99%.

Finally, we present exhaustive top-1 classification accuracy results on the CIFAR-10 dataset for a variety of gradient distortion attacks coupled with behavior patterns of the adversarial nodes. Our results indicate an average 30% accuracy increase on CIFAR-10 [19] under the most sophisticated attacks.

II. Distributed Training Formulation

As in typical distributed learning setups, we assume a loss function $l_i(\mathbf{w})$ for the i^{th} sample where $\mathbf{w} \in \mathbb{R}^d$ is the

This work was supported in part by the National Science Foundation (NSF) under grant CCF-1910840 and grant CCF-2115200.

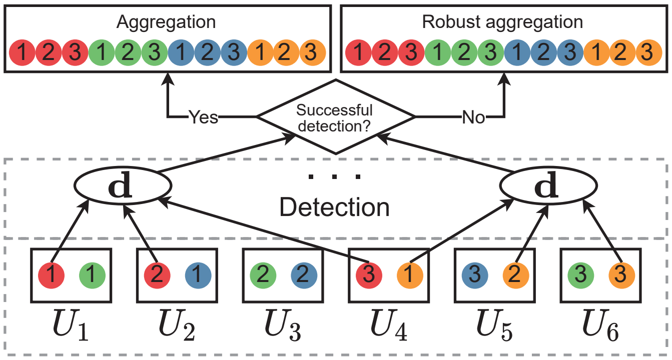


Fig. 1: Aggregation of gradients on a cluster.

parameter set of the model.[1] We use *mini-batch SGD* to minimize the loss over the entire dataset, i.e.,

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

where n is the dataset size. Initially, $\mathbf{w}$ is randomly set to $\mathbf{w}_0$ ($\mathbf{w}_t$ is the model at the end of iteration t). A random *batch* B_t of b samples is chosen for the update in the t^{th} iteration

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

where η_t is the learning rate of 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 dataset and the model and coordinates the protocol.

Task assignment: Each batch B_t is split into f disjoint files $\{B_{t,i}\}_{i=0}^{f-1}$, which are then assigned to the workers according to our placement policy. Redundancy is introduced by assigning a given file to $r > 1$ workers. Each worker is responsible for $l = fr/K$ files (l is the *computation load*). 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 file $B_{t,i}$. In Aspis, $\mathcal{N}^f(B_{t,i})$ uniquely identifies the file $B_{t,i}$; thus, we will sometimes refer to the file $B_{t,i}$ by its *group* of assigned workers, $\mathcal{N}^f(B_{t,i})$. The placement algorithm will be presented in Section III.

Adversary model: We assume that at most q workers can be adversarial. The workers know the data assignment of all nodes, the parameters $\mathbf{w}_t$, and the defense at *every* iteration (*omniscient* attack); they can also collude. The adversarial machines may change at every single iteration. We will suppose that $q < K/2$. We emphasize that our attack setting is more powerful than random failures considered in related redundancy-based work [14], [15]. For each assigned file $B_{t,i}$ a worker U_j will return the value $\hat{\mathbf{g}}_{t,i}^{(j)}$ to the PS. Then,

$$\hat{\mathbf{g}}_{t,i}^{(j)} = \begin{cases} \mathbf{g}_{t,i} & \text{if } U_j \text{ is honest,} \\ * & \text{otherwise,} \end{cases} \tag{2}$$

where $\mathbf{g}_{t,i}$ is the following sum of loss gradients

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

and $*$ is any arbitrary vector in $\mathbb{R}^d$.

Training: We will refer to Figure 1 for this exposition. There are $K = 6$ machines and $f = 4$ distinct files (colored circles) replicated $r = 3$ times each.[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 detection operations the PS performs after receiving all the gradients.

The algorithm starts with the assignment of files to workers. Subsequently, each worker U_i will compute all l file gradients that involve its assigned files $\mathcal{N}^w(U_i)$ and return them to the PS. In every iteration, the PS will initially run our detection algorithm in an effort to identify the q adversaries and will act differently depending on the detection outcome.

- *Case 1: Successful detection*. The PS will ignore all detected faulty machines and keep only the gradients from the remaining workers. 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. (1)). 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 an honest copy, i.e., either of U_5 or U_6.
- *Case 2: Unsuccessful detection*. During aggregation, the PS will perform a majority vote across the computations of each file. Recall that each file has been processed by r workers. For each such file $B_{t,i}$, the PS decides a majority value $\mathbf{m}_i$

$$\mathbf{m}_i := \text{majority}\left\{\hat{\mathbf{g}}_{t,i}^{(j)} : U_j \in \mathcal{N}^f(B_{t,i})\right\}. \tag{3}$$

Assume that r is odd and let $r' = \frac{r+1}{2}$. Under the rule in Eq. (3), the gradient on a file is distorted only if at least r' of the computations are corrupted. Following the majority vote, we will further filter the gradients using coordinate-wise median and refer to the combination of these two steps as *robust aggregation*. For example, in Figure 1, all returned values for the red file will be evaluated by majority voting on the PS, which decides a single output value; the same is done for the other 3 files. After voting, Aspis applies coordinate-wise median on the "winning" gradients $\mathbf{m}_i$, $i = 0, 1, \ldots, f-1$.

The procedural details are described in [20, Algorithm 1].

Metrics: Our main metrics are the fraction of distorted files and the top-1 test accuracy of the trained model. We evaluate these metrics for the various competing methods.

III. Task Assignment

In this section, we propose our technique which determines the allocation of gradient tasks to workers. If $\mathcal{U}$ is the set of workers, Aspis has $|\mathcal{U}| \leq f$ (i.e., fewer workers than files). To allocate the batch of an iteration, B_t, to the K workers, first, we will partition B_t into $f = \binom{K}{r}$ disjoint files $B_{t,0}, B_{t,1}, \ldots, B_{t,f-1}$; recall that r is the redundancy. Following this, we associate each file with exactly one of the subsets $S_0, S_1, \ldots, S_{\binom{K}{r}-1}$ of $\{U_1, U_2, \ldots, U_K\}$ each of cardinality r, essentially using a bijection. Each file contains b/f samples. The details are specified in Algorithm 1, and the following example showcases its protocol.

[1] The paper's heavily-used notation is summarized in [20, Table 6].

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

Algorithm 1: Aspis subset-based file assignment.

```
Input: Batch size $b$, computation load $l$, redundancy $r$
       and worker set $\mathcal{U}$, $|\mathcal{U}| = K$.
PS partitions batch $B_t$ into $f = \binom{K}{r}$ disjoint files of
  $b/f$ samples each
      $B_t = \{B_{t,i} : i = 0, 1, \ldots, f-1\}$.
PS constructs all subsets $S_0, S_1, \ldots, S_{\binom{K}{r}-1}$ of
  $\mathcal{U} = \{U_1, U_2, \ldots, U_K\}$ such that $\forall i, |S_i| = r$.
for $i = 0$ to $f-1$ do
    PS identifies all workers in group
      $S_i = \{U_{j_1}, U_{j_2}, \ldots, U_{j_r}\}$ and assigns the file
      indexed with $i$ in $B_t$ to all of them. Formally,
      $\mathcal{N}^w(U_j) = \mathcal{N}^w(U_j) \cup \{B_{t,i}\}$ for $j \in \{j_1, \ldots, j_r\}$.
end
```

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 $S_0 = \{U_1, U_2, U_3\}$ corresponds to file $B_{t,0}$ and will be processed by U_1, U_2, and U_3.

IV. Adversarial Detection

The PS will run our detection method in every iteration as our model assumes that the adversaries can be different across different steps; for brevity, the iteration index t will be omitted from most of the notation. 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. 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 an iteration, let us encode the agreement of workers U_{j_1}, U_{j_2} on a common file i of them as

$$\alpha_i^{(j_1,j_2)} := \begin{cases} 1 & \text{if } \hat{\mathbf{g}}_i^{(j_1)} = \hat{\mathbf{g}}_i^{(j_2)}, \\ 0 & \text{otherwise}. \end{cases} \tag{4}$$

Then, across all files, let us denote the total number of agreements between a pair of workers U_{j_1}, U_{j_2} by

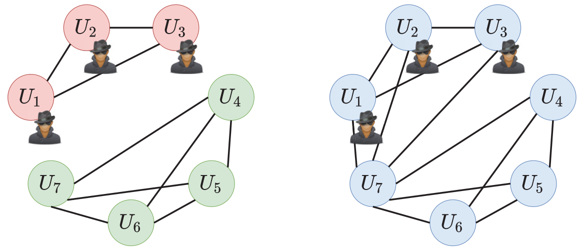
$$\alpha^{(j_1,j_2)} := \sum_{i \in \mathcal{N}^w(U_{j_1}) \cap \mathcal{N}^w(U_{j_2})} \alpha_i^{(j_1,j_2)}. \tag{5}$$

Since the placement is known, the PS can always perform the above computation. Next, we form an undirected graph $\mathbf{G}$ whose vertices correspond to all workers $\{U_1, U_2, \ldots, U_K\}$. An edge (U_{j_1}, U_{j_2}) exists in $\mathbf{G}$ only if the computed gradients of U_{j_1} and U_{j_2} match in all their $\binom{K-2}{r-2}$ common groups.

A *clique* in an undirected graph is defined as a subset of vertices in which there is 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. The set of honest workers H will pair-wise agree everywhere; hence, the subset 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 straight away be eliminated as a "detected" adversary.

Algorithm 2: 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
```



(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.

The essential idea of our detection is to run a clique-finding algorithm on $\mathbf{G}$ (Algorithm 2). If we find a unique maximum clique, we declare it to be the set of honest workers; the gradients from the detected adversaries are ignored. If there is more than one maximum clique, we resort to the robust aggregation discussed in Section II. Let us denote the number of distorted tasks upon Aspis detection and aggregation by $c^{(q)}$ and its maximum value (under the worst-case attack) by $c_{\max}^{(q)}$. The *distortion fraction* is $\epsilon := c^{(q)}/f$. Clique-finding is an NP-complete problem [21]. Nevertheless, there are fast, practical algorithms with excellent performance on graphs even up to hundreds of nodes [22], [23]. We utilize the algorithm of [23] which is optimal in enumerating all maximal cliques. Our extensive experimental evidence suggests that clique-finding is not a bottleneck for the size and structure of the graphs that Aspis uses, and even for $K = 100$ workers and $r = 5$ took approximately 15 milliseconds. In [20, Section A.1], the asymptotic complexity of the entire protocol is discussed.

A. Weak Adversarial Strategy

We first consider a class of weak attacks where the Byzantine nodes attempt to distort the gradient on any file they participate in. It is clear that each Byzantine node will disagree with at least $K - q$ honest nodes, 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. The algorithm will detect all adversaries, declare the (unique) maximum clique as honest, and proceed to aggregation (*cf.* Section II). The only files that can be distorted are those that consist exclusively of adversaries.

Figure 2a (corresponding to Example 1) shows 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 $H = \{U_4, U_5, U_6, U_7\}$. The unique maximum clique is $M_{\mathbf{G}} = H$, and detection is successful. Under this attack, the distorted tasks are those whose all copies have been compromised, i.e., $c^{(q)} = \binom{q}{r}$.

B. Optimal Adversarial Strategy

Our second scenario is strong and involves adversaries which collude in the "best" way possible while knowing the full details of our algorithm. We provide an upper bound on the number of files that can be corrupted and demonstrate a strategy that the adversarial workers can follow to achieve this upper bound, also referred to as an *optimal* strategy.

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 on *at least one* of the $\binom{K-2}{r-2}$ files that they jointly participate in. To corrupt the gradients, each adversary has to disagree on the computations with a subset of the honest workers. 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 H.

Upon the formation of $\mathbf{G}$, a worker U_j will be flagged as adversarial if $deg(U_j) < K - q - 1$. Therefore to avoid detection, a *necessary* condition is $|D_j| \leq q$. We fall back to robust aggregation in case of more than one maximum clique in $\mathbf{G}$. Then, a gradient can only be corrupted if most of its assigned workers are adversarial and agree on a wrong value.

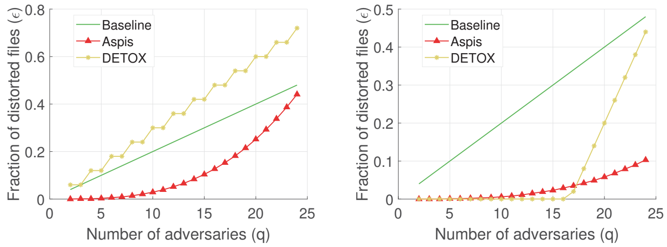
For a given file F, let $A' \subseteq A$ with $|A'| \geq r'$ be the set of "active adversaries" in it, i.e., $A' \subseteq F$ consists of Byzantines that collude to create a majority that distorts its gradient. The remaining workers in F belong to $\cap_{i \in A'} D_i$, where $|\cap_{i \in A'} D_i| \leq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

$$\begin{aligned} X_j \quad = \quad & \{F : \exists A' \subseteq A \cap F, |A'| = j, \\ & \text{and } \forall\ U_j \in F \setminus A', U_j \in \cap_{i \in A'} D_i\}. \end{aligned} \tag{6}$$

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.,

$$c_{\max}^{(q)} = \max_{D_i, |D_i| \leq q, i=1,2,\ldots,q} | \cup_{j=r'}^{r} X_j| \tag{7}$$

where the maximization is over the choices of the disagreement sets $D_1, D_2, \ldots, D_q$. An intuitive strategy based on Eq. (6) is to maximize the set $\bigcap_{i \in A' \cap F} D_i$ for every possible file F; for all groups F, the adversaries need to fix a subset of q non-adversaries, say $D \subset H$, to be the set of workers with which all adversaries will disagree, i.e., $D_i = D$ for $i = 1, 2, \ldots, q$. We present our main theorem (proved in [20, Section A.2]).



(a) Optimal attacks.

(b) Weak attacks.

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

Theorem 1. In a training cluster of K workers and q adversaries using Algorithm 1 to assign the $f = \binom{K}{r}$ files, and Algorithm 2 for detection, an optimal adversary model can corrupt at most

$$c_{\max}^{(q)} = \frac{1}{2}\binom{2q}{r} \tag{8}$$

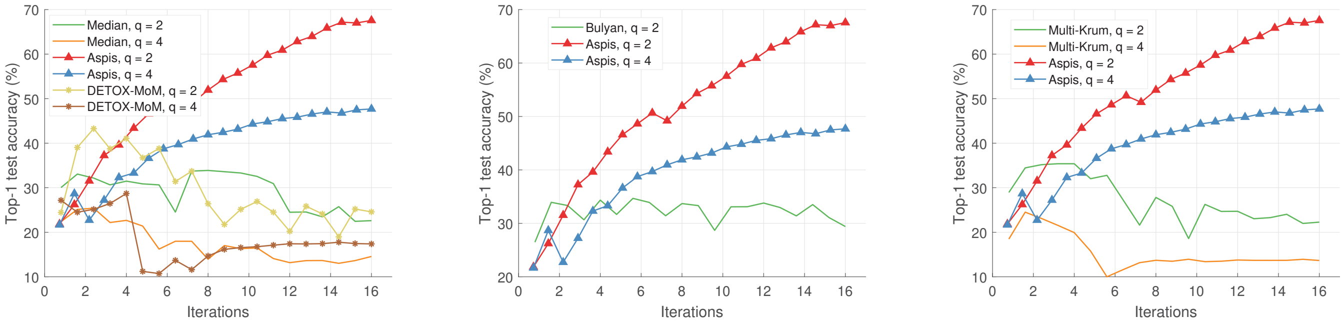
files. Furthermore, this upper bound can be achieved if all adversaries fix a set $D \subset H$ of honest workers with which they will consistently disagree on the gradient (by distorting it).

One such attack is carried out in Figure 2b for 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.

V. Distortion Fraction Evaluation

We have performed simulations of the fraction of distorted files (defined as $\epsilon = c^{(q)}/f$) incurred by Aspis and competing aggregators. The main motivation is that our deep learning experiments (*cf.* Section VI-B) as well as our prior work [12] show that ϵ serves as a surrogate of the model's convergence with respect to accuracy. In addition, our simulations show that Aspis enjoys values of ϵ, which are as much as 99% lower for the same q compared to other techniques, and this attests to our theoretical robustness guarantees. 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 compare our work with *baseline* approaches that do not involve redundancy or majority voting. Their aggregation is applied directly to the K gradients returned by the workers ($f = K$, $c_{\max}^{(q)} = q$ and $\epsilon = q/K$).

Let us discuss *optimal* attacks. For Aspis, we used the proposed attack from Section IV-B and the corresponding computation of $c^{(q),Aspis}$ of Theorem 1. *DETOX* in [14] employs redundancy followed by majority voting and offers robustness guarantees which crucially rely on a "random choice" of the Byzantines. In prior work [12], we have observed that redundancy is not sufficient to yield Byzantine resilience dividends and proposed an optimal choice of the q Byzantines that maximizes ϵ^{DETOX}, which we used in our experiments and incurs $c^{(q),DETOX} = \lfloor \frac{q}{r'} \rfloor$ and $\epsilon^{DETOX} = \lfloor \frac{q}{r'} \rfloor \times r/K$. We also compare with the distortion fraction of ByzShield [12] 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 options. The reader can refer to Figure



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

Fig. 4: *ALIE* distortion under optimal attack scenarios (CIFAR-10), $K = 15$.

3a and [20, Tables 1,2,3] for our results. Aspis achieves major reductions in ϵ; for instance, ϵ^{Aspis} is reduced by up to 99% compared to both $\epsilon^{Baseline}$ and ϵ^{DETOX} in Figure 3a.

We will next introduce the utilized *weak* attacks (*cf.* Figure 3b). For our scheme, we will make an arbitrary choice of q adversaries which carry out the method introduced in Section IV-A, i.e., they will distort all files, and a successful detection is possible. As discussed, the fraction of corrupted gradients is $\epsilon^{Aspis} = \binom{q}{r}/\binom{K}{r}$ in that case. For DETOX, a simple benign attack is used. Let the K/r files of DETOX be $B_{t,0}, B_{t,1}, \ldots, B_{t,K/r-1}$ (*cf.* [14]). 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 $B_{t,i \mod K/r}$ to the adversarial set A. Then,

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

VI. Large-scale Deep Learning Experiments

A. Experiment Setup

We evaluated the performance of all compared techniques on Amazon EC2. The project is written in PyTorch [24] and uses the MPICH library for communication. We worked with the CIFAR-10 dataset [19] using the ResNet-18 [25] model. We used clusters of $K = 15$ and 21 workers and redundancy $r = 3$. We simulated values of $q = 2, 4, 6$ during training.

There are two different adversarial dimensions we evaluate:
1) Choice/orchestration of the adversaries: the different ways adversaries are chosen and work together to inflict damage (*cf.* weak and optimal attacks in Section IV).
2) Gradient distortion methods: the method an adversary uses to distort the gradient value. We use a variety of state-of-the-art methods for distorting the computed gradients, including *ALIE* [26], *Fall of Empires (FoE)* [27] and *reversed gradient*, which returns $-c\mathbf{g}$, $c > 0$ instead of the true gradient $\mathbf{g}$.

Competing methods: We compare against the baseline implementations of median-of-means [28], Bulyan [11], and Multi-Krum [6]. If $c^{(q)}_{\max}$ is the number of adversarial computations, then Bulyan and Multi-Krum require at least $4c^{(q)}_{\max} + 3$ and $2c^{(q)}_{\max} + 3$ total number of computations, respectively. These constraints make these methods inapplicable for larger values of q. We also compare with DETOX [14], which can easily fail under malicious scenarios for large q. We compare with median-based techniques since they originate from robust statistics and are the base for many aggregators. DETOX is the most related redundancy-based work that is based on coding theory. Finally, Multi-Krum is a highly-cited aggregator that combines majority-based and squared-distance-based methods.

B. Experimental Results

1) Comparison under optimal attacks: Figure 4a 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).[3] In Figures 4b and 4c, 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 $K = 15 < 4c^{(q)}_{\max} + 3 = 4q + 3 = 19$. Also, neither Bulyan nor Multi-Krum can be paired with DETOX for $q \geq 1$ since the inequalities $f \geq 4c^{(q)}_{\max} + 3$ and $f \geq 2c^{(q)}_{\max} + 3$ cannot be satisfied (*cf.* [6], [11]). Also, note that the accuracy of most competing methods fluctuates more than in the results presented in the corresponding papers [14] and [26]. This is expected as we consider stronger attacks than those papers, i.e., optimal deterministic attack on DETOX and, in general, up to 27% adversarial workers in the cluster. The results for the reversed gradient attack are shown in [20, Figures 5a, 5b, 5c]. Given that this is a much weaker attack [12], [14], all schemes are expected to perform well. Under the Fall of Empires (FoE) distortion (*cf.* [20, Figure 6]) 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.

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 at most q out of the K gradients. Hence, for weak scenarios, we chose to compare mostly with DETOX. The accuracy is reported in [20, Figures 7, 8] according to which Aspis shows an improvement under attacks on the more challenging end of the spectrum (ALIE). The results are in line with the theoretical distortion fraction, which is $\epsilon^{Aspis} = 0.044$ while $\epsilon^{Baseline} = 0.4$ and $\epsilon^{DETOX} = 0.2$ for $q = 6$.

Our experiments on a larger cluster of $K = 21$ workers under the ALIE attack can be found in [20, Figure 9].

[3] Please refer to [20, Tables 1, 2] for the values of the distortion fraction ϵ each scheme incurs.

References

[1] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, June 2014, pp. 361—372.

[2] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019, pp. 1211–1220.

[3] N. Gupta and N. H. Vaidya, "Byzantine fault-tolerant parallelized stochastic gradient descent for linear regression," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, September 2019, pp. 415–420.

[4] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *Advances in Neural Information Processing Systems*, December 2018, pp. 4618–4628.

[5] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, December 2017.

[6] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, December 2017, pp. 119–129.

[7] G. Damaskinos, E. M. El Mhamdi, R. Guerraoui, A. H. A. Guirguis, and S. L. A. Rouault, "AggreGathor: Byzantine machine learning via robust gradient aggregation," in *Conference on Systems and Machine Learning (SysML)*, March 2019, p. 19.

[8] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Defending against saddle point attack in Byzantine-robust distributed learning," in *Proceedings of the 36th International Conference on Machine Learning*, June 2019, pp. 7074–7084.

[9] ——, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*, July 2018, pp. 5650–5659.

[10] C. Xie, O. Koyejo, and I. Gupta, "Generalized Byzantine-tolerant SGD," March 2018. [Online]. Available: https://arxiv.org/abs/1802.10116

[11] E. M. El Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proceedings of the 35th International Conference on Machine Learning*, July 2018, pp. 3521–3530.

[12] K. Konstantinidis and A. Ramamoorthy, "ByzShield: An efficient and robust system for distributed training," in *Machine Learning and Systems 3 (MLSys 2021)*, April 2021.

[13] 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," April 2019. [Online]. Available: https://arxiv.org/abs/1806.00939

[14] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, "DETOX: A redundancy-based framework for faster and more robust gradient aggregation," in *Advances in Neural Information Processing Systems*, December 2019, pp. 10 320–10 330.

[15] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "DRACO: Byzantine-resilient distributed training via redundant gradients," in *Proceedings of the 35th International Conference on Machine Learning*, July 2018, pp. 903–912.

[16] D. Data, L. Song, and S. N. Diggavi, "Data encoding for Byzantine-resilient distributed gradient descent," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, October 2018, pp. 863–870.

[17] J. Regatti, H. Chen, and A. Gupta, "ByGARS: Byzantine SGD with arbitrary number of attackers," December 2020. [Online]. Available: https://arxiv.org/abs/2006.13421

[18] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *Proceedings of the 36th International Conference on Machine Learning*, June 2019, pp. 6893–6901.

[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[20] K. Konstantinidis and A. Ramamoorthy, "Aspis: Robust detection for distributed learning," January 2022. [Online]. Available: https://arxiv.org/abs/2108.02416

[21] R. M. Karp, *Reducibility among Combinatorial Problems*. Boston, MA: Springer US, 1972.

[22] F. Cazals and C. Karande, "A note on the problem of reporting maximal cliques," *Theoretical Computer Science*, vol. 407, no. 1, pp. 564–568, November 2008.

[23] T. Etsuji, T. Akira, and T. Haruhisa, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theoretical Computer Science*, vol. 363, no. 1, pp. 28–42, October 2006.

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, December 2019, pp. 8024–8035.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[26] G. Baruch, M. Baruch, and Y. Goldberg, "A Little Is Enough: Circumventing defenses for distributed learning," in *Advances in Neural Information Processing Systems*, December 2019, pp. 8635–8645.

[27] C. Xie, O. Koyejo, and I. Gupta, "Fall of Empires: Breaking byzantine-tolerant sgd by inner product manipulation," in *35th Conference on Uncertainty in Artificial Intelligence, UAI 2019*, July 2019, pp. 6893–6901.

[28] S. Minsker, "Geometric median and robust estimation in Banach spaces," *Bernoulli*, vol. 21, no. 4, pp. 2308–2335, November 2015.