Attacking Byzantine Robust Aggregation in High Dimensions

Sarthak Choudhary*, Aashish Kolluri* and Prateek Saxena
School of Computing, National University of Singapore
csarthak76@gmail.com {aashish7, prateeks}@comp.nus.edu.sg

***Abstract*—Training modern neural networks or models typically requires averaging over a sample of high-dimensional vectors. Poisoning attacks can skew or bias the average vectors used to train the model, forcing the model to learn specific patterns or avoid learning anything useful. Byzantine robust aggregation is a principled algorithmic defense against such biasing. Robust aggregators can bound the maximum bias in computing centrality statistics, such as mean, even when some fraction of inputs are arbitrarily corrupted. Designing such aggregators is challenging when dealing with high dimensions. However, the first polynomial-time algorithms with strong theoretical bounds on the bias have recently been proposed. Their bounds are *independent* of the number of dimensions, promising a conceptual limit on the power of poisoning attacks in their ongoing arms race against defenses.**

In this paper, we show a new attack called HIDRA on practical realization of strong defenses which subverts their claim of dimension-independent bias. HIDRA highlights a novel computational bottleneck that has not been a concern of prior information-theoretic analysis. Our experimental evaluation shows that our attacks almost completely destroy the model performance, whereas existing attacks with the same goal fail to have much effect. Our findings leave the arms race between poisoning attacks and provable defenses wide open.

1. Introduction

Machine learning training algorithms often have to compute an average over a set of vectors. The standard stochastic gradient descent (SGD) algorithm for neural network training, as an example, computes the average of gradient vectors derived from data samples in each step. Similarly, in federated learning, local models are trained at individual worker machines and then sent to a central service that averages over these vectors to get the global model.

Several security problems arise if some of these vectors can be maliciously crafted. For instance, poisoning attacks corrupt training data samples so that gradients computed from them during SGD skew or bias the learned model. The *bias* is how far the average computed from the partially corrupted vectors can be from the benign (uncorrupted) value. Such biasing attacks on ML training can severely deteriorate the training accuracy of the model [1], [2], [3], exacerbate privacy concerns [4], and create unfairness that did not exist in the dataset [5], [6]. Local models sent by malicious worker machines in federated learning can directly be corrupted before being sent to the central service.

A natural security question arises about the robustness of such averaging. It is easy to see that even a single corrupted input vector can result in arbitrarily biasing the average when taking a simple arithmetic mean. It is desirable to design algorithms that compute an *aggregate* statistic over given vectors, similar to the mean, which cannot be biased much even by a strong adversary. The *Byzantine Robust Aggregation* problem captures this goal: Imagine a fraction ϵ of vectors can be arbitrarily corrupted, then can we ensure the bias in the computed aggregate is small [7], [8]? Robust aggregation algorithms bound how much the attacker can skew the average model at each step in training, thereby offering a principled limit on the effect of attacks.

Designing robust aggregators is an algorithmic challenge that has inspired many attempts. These aggregators, as shown in Table 1, most often provide a theoretical upper bound [1] on the maximum bias of $O(\sqrt{\epsilon d})$ [9], [10], [11], [12]. The bias is the L_2 norm of the maximum adversarial error induced, ϵ is the fraction of vectors controlled by the adversary, and d is the number of dimensions of the vectors. The issue in practice is that the number of dimensions d is usually the parameter size of the ML model, which can be in millions or billions. The bias bound being dependent on d makes it vacuously large for modern ML systems. Hence, we refer to these algorithms as weakly bounded (or *weak*). Concrete poisoning attacks that have a severe impact on weak algorithms have been shown recently [2], [3], [13].

On the theoretical side, robust aggregation or robust mean estimation has been a long-standing challenge in high-dimensional statistics. The primary goal is to give *stronger bounds of $O(\sqrt{\epsilon})$ on the bias*, without the $\sqrt{d}$ factor, which is the statistically optimal [14], [15]. The classical Tukey median [8] achieves the goal but best known algorithms for computing it have running time exponential in d [16]. A new line of algorithms gives the first polynomial time solutions with strong bounds on bias [17] and their first practical realization strategy is shown recently [14]. Table 1 summarizes the guarantees of strong aggregators. It has been experimentally verified that they completely thwart existing poisoning attacks which aim to reduce the trained model performance, thus establishing the state-of-the-art [14]. Im-

*. These authors contributed equally to this work.

1. The more precise asymptotical bounds are in Table 1. We will often drop the Σ term, as it is a constant for a given distribution.

TABLE 1: Comparison of the worst-case bias between different robust aggregation algorithms. $||\Sigma||_2$ is maximum variance of the uncorrupted sample. $\tilde{O}(\cdot)$ ignores the constant and logarithmic factors in the computation complexity. The number of vectors is n and dimensions d.

Algorithm	Max. Bias	Comp. Complexity
Weak robust aggregators		
Median [9]	$\tilde{O}(\sqrt{\epsilon d}) \cdot ||\Sigma||_2^{\frac{1}{2}}$	$\tilde{O}(nd)$
Trimmed Mean [9]	$\tilde{O}(\sqrt{\epsilon d}) \cdot ||\Sigma||_2^{\frac{1}{2}}$	$\tilde{O}(nd)$
Geometric Median [10], [11], [12]	$\tilde{O}(\sqrt{\epsilon d}) \cdot ||\Sigma||_2^{\frac{1}{2}}$	$\tilde{O}(nd)$
Strong robust aggregators		
Filtering [17]	$\tilde{O}(\sqrt{\epsilon}) \cdot ||\Sigma||_2^{\frac{1}{2}}$	$\tilde{O}(\epsilon n \cdot d^3)$
No-Regret [18]	$\tilde{O}(\sqrt{\epsilon}) \cdot ||\Sigma||_2^{\frac{1}{2}}$	$\tilde{O}((n+d^3) \cdot d)$
SoS [19]	$\tilde{O}(\sqrt{\epsilon}) \cdot ||\Sigma||_2^{\frac{1}{2}}$	poly(n, d)
Tukey Median [8]	$\tilde{O}(\sqrt{\epsilon}) \cdot ||\Sigma||_2^{\frac{1}{2}}$	NP-Hard in d

portantly, these are the only polynomial-time algorithms known to have strong bounds on the bias independent of d—a crucial security property when working with high-dimensional vectors as in modern ML training.

Our work. Since none of the existing attacks defeat strong robust aggregators, even when working in low dimensions, it is natural to ask: Are there *any* attacks that create bias matching the theoretical upper bound of $O(\sqrt{\epsilon})$ in them? In this paper, we present the first effective attack against strong robust aggregators, called HIDRA[2]. HIDRA induces bias matching their analytical upper bounds in low dimensional settings. Our attack, thus, shows that the prior theoretical bounds for strong defenses are tight.

More importantly, we observe that the bounds given by prior theoretical analysis make *idealized computational assumptions* which hold primarily when the number of dimensions is small. As the number of dimensions increases, existing robust aggregators run into a fundamental computation bottleneck. Practical realizations of these defenses, therefore, when working with high dimensions, have to break down the given vectors into multiple low-dimensional chunks to solve for. Our HIDRA attack induces a near optimal[3] bias per chunk, resulting in a total bias of $\Omega(\sqrt{\epsilon d})$ in the high dimensional setting. *This is in sharp contrast to the promised dimension independent $\tilde{O}(\sqrt{\epsilon})$ bias*. A factor of $\sqrt{d}$ translates to several orders of magnitude worse bias, even for moderately sized neural networks that have a million parameters. We analytically derive the lower bound on the bias achieved by our HIDRA attack on the above chunking procedure of $\Omega(\sqrt{\epsilon d})$ and experimentally confirm that corruption of input vectors using HIDRA hits this lower bound. HIDRA is thus, again, near-optimal for high d.

Experimental results. We employ HIDRA towards creating indiscriminate or untargeted poisoning attacks as a concrete application [20]. Untargeted poisoning attacks aim to prevent ML models from learning useful information and have been difficult to achieve with prior attacks when strong robust aggregators are used. However, our attack consistently results in a drastic drop in the accuracy of trained models even when using strong robust aggregators. For example, in several instances **the original model accuracy of over 80% drops to below 10% with $\epsilon = 0.2$ fraction of vectors corrupted** using HIDRA. Under the same setup, prior attacks induce *below* 5% drop in performance.

Computational Bottleneck is fundamental. The computational bottleneck targeted by HIDRA is fundamental to the problem of robust aggregation in general, not specific to a single algorithm. The key idea in all strong robust aggregators is to filter out vectors far from the mean in the *direction of maximum variance* of the given vectors. For this, strong aggregators compute the maximum variance direction. It turns out that this is no accident. We show a quasilinear time reduction from the problem of computing the maximum variance direction approximately to that of strong robust aggregation, for a non-trivial class of inputs (see Section 4.4). In other words, solving strong aggregation efficiently would imply finding the approximate maximum variance direction efficiently for this class of inputs.

To our knowledge, the best-known algorithms for computing the maximum variance direction of given vectors have $O(d^3)$ [4] time complexity [21], [22] Their highly optimized implementations[5] remain $O(d^3)$, despite being widely used for decades. For modern ML models, d can be 2^{20} to 2^{40}, so $O(d^3)$ is prohibitively costly. The computational bottleneck in strong aggregators, therefore, does not appear easy to work around in full generality.

Our contribution. We present the first effective attacks against provable byzantine robust aggregation defenses that have the strongest bias bounds. Our attacks are *near optimal* and experimentally surpass the efficacy of existing untargeted poisoning attacks for these defenses. They create a severe loss in performance of trained models. The vulnerability is *computational*, highlighting the gaps between idealized information-theoretic analysis of these defenses and their practical realization. As a result, designing practical and provable defenses that limit the bias in poisoning attacks remains an open question for the future.

2. Background & Problem Setup

The central question we are interested in is that of computing a "mean-like" aggregate statistic of n vectors when a small unknown fraction ϵ of them has been arbitrarily corrupted. The defender wants the computed aggregate to be *robust*, i.e., not too far from the mean of uncorrupted

2. HIDRA is short for **Hi**gh **D**imensional attack on **R**obust **A**ggregators.

3. The known upper bound is $\tilde{O}(\sqrt{\epsilon d})$, see Table 1.

4. Algorithms for exact maximum variance run in $O(d^3)$, while the approximate algorithms with desired error rates operate in $\tilde{O}(n^2 d)$. We refer readers to section 4.4 for more details.

5. https://numpy.org/doc/stable/reference/generated/numpy.linalg.eigh.html

vectors. The problem arises in many applications, though we are motivated by its use in machine learning tasks.

2.1. The Problem: Byzantine Robust Aggregation

There are n vectors $X = \{x_1, ..., x_n\}$, for example n gradients of a neural network, where each vector $x_i \in \mathbb{R}^d$. A *Byzantine* adversary who has access to X can replace an $\epsilon < \frac{1}{2}$ fraction of vectors in X and with arbitrarily chosen vectors, resulting in a set $Y = \{y_1, ..., y_n\}$ called the ϵ-corrupted set. The defender is given Y and some auxiliary information about X. The robust aggregation problem is to specify an aggregation function $f : R^{n \times d} \to \mathbb{R}^d$, such that the *bias*—the difference between $f(Y)$ and the arithmetic mean of X for all possible choices of Y—is bounded by λ. Formally, there exists some $\lambda > 0$ such that:

$$bias = \max_{Y} \| f(Y) - \frac{1}{n} \sum_{i=1}^{n} x_i \|_2 \leq \lambda \tag{1}$$

There are known information-theoretic limits for which sets we can solve robust aggregation. Solving the above problem with provable guarantees in full generality, without making any assumptions about the *uncorrupted* samples X, is not possible. One must minimally assume that an upper bound on the variance of uncorrupted samples is known [17], [19], [23]—without it, there exist sets where a small number of corruptions make it impossible to recover the original mean. We will therefore assume that a reasonably good estimate of maximum variance is known in advance. The maximum variance is the *largest eigenvalue* or the *spectral norm* of Σ, denoted by $||\Sigma||_2$, where $\Sigma \in \mathbb{R}^{d \times d}$ is the covariance matrix of X defined below:

$$\Sigma = \frac{1}{n} \cdot \left[\sum_{i=0}^{n} (x_i - \hat{\mu})^T \cdot (x_i - \hat{\mu}) \right]; \hat{\mu} = \frac{1}{n} \sum_{i=0}^{n} x_i$$

The statistically optimal (best possible) bias λ is a multiplicative factor, independent of d, times the maximum variance $||\Sigma||_2^{\frac{1}{2}}$. We refer to this multiplicative factor as the *bound* when clear from context. $||\Sigma||_2^{\frac{1}{2}}$ is fixed by the distribution X is drawn from, but it may depend on d.

The robust aggregation problem is also called robust mean estimation in statistics [8], [15], [18], [19], [23], [24], [25], [26], [27]. The above assumption about knowing $||\Sigma||_2$ is standard in prior work on robust mean estimation [17].

2.2. Application: Untargeted Poisoning in SGD

The standard training algorithm for neural networks is stochastic gradient descent (SGD) [28]. We will use training with SGD as the baseline throughout the paper.

Aggregation in SGD. SGD works by initializing the ML model parameters or weights $w \in \mathbb{R}^d$ with random values and iteratively updating it in sequential steps. In each step t, it selects a set of n samples uniformly at random from the training dataset and computes the aggregate:

$$w^{t+1} = w^t - \frac{\eta}{n} \sum_{i=1}^{n} \nabla Q_i(w^t)$$

Here, the number of samples taken in each set is a constant n, the gradient of the i^{th} data point in the set is $\nabla Q_i(w)$, and the learning rate η is a constant. The gradients are vectors in $\mathbb{R}^d$. The choice of loss functions, neural network architecture, and so on are not relevant for the purpose of modeling the problem of robust aggregation.

The vanilla SGD procedure, as described above, is susceptible to adversarial corruptions of gradients. This can result in arbitrary bias since a simple arithmetic mean (average) of gradients is computed in each training step. A more robust aggregation technique than arithmetic mean is a natural mitigation for attacks that try to bias the aggregate.

Threat model. We assume that the defender will use a strong robust aggregator f instead of naive arithmetic mean during the SGD computation. These aggregators take as input the corrupted samples Y and the auxiliary information $||\Sigma||_2$ about the benign samples. In practice, one can estimate $||\Sigma||_2$ experimentally based on a small set of clean unpoisoned samples. We assume the defender knows $||\Sigma||_2$ to use in f, and the attacker also knows f and $||\Sigma||_2$.

The Byzantine attacker model admits a powerful attacker that can arbitrarily corrupt vectors being aggregated. Such an attack model abstracts away from how the attacker induces the corruption, for instance, whether through poisoning of datasets [20] or direct control over the vectors [29]. In typical centralized training, we are concerned with attackers who control training data samples and can poison them. Gradients computed from poisoned samples are thus influenced significantly by the attacker but not necessarily fully controlled. In contrast, certain federated learning setups offer more direct control over gradient vectors being aggregated. In federated learning, a server collaborates to train a machine learning model with several clients or worker machines. Clients compute local updates (vectors) to the model on their local datasets and send them to the server, which aggregates them as in vanilla SGD. It is easy to see that a compromised or malicious worker has direct and complete control over the vector they send. Our Byzantine attacker model thus captures both the centralized and federated learning setups.

The adversary is *fully adaptive*, meaning that they corrupt with complete knowledge of the function f and $||\Sigma||_2$ in every training round. The ϵ-corrupted sample of X, denoted as Y, is given to the robust aggregator (defender).

Untargeted model poisoning attacks. While there can be many attacker goals of biasing models during training, we focus solely on one type of attacks, namely *untargeted poisoning* [20], to show practical value. The attack corrupts the gradient vectors computed during training. The primary objective of untargeted poisoning is to bias the aggregated model at each step such that it exhibits a high error rate for

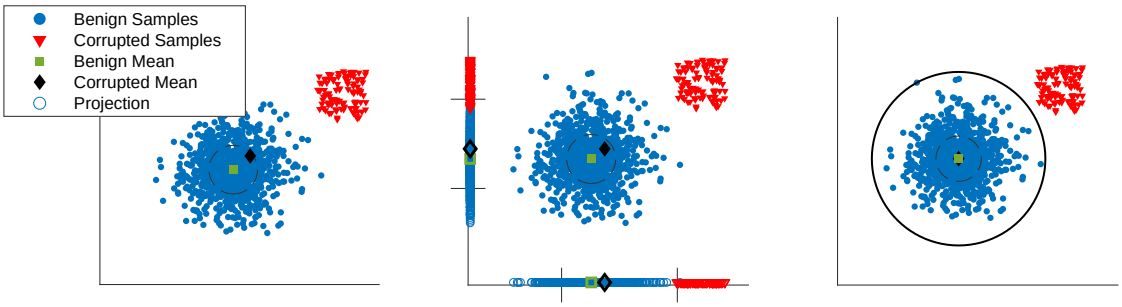

Figure 1: Left: Gaussian samples with 0.1 (ϵ) fraction of arbitrarily corrupted data, highlighting the mean shift post-corruption where the dotted circle is the standard deviation (σ) boundary around the mean. Middle: Trimmed mean by dimension, establishing dimension-wise thresholds to contain corrupted mean within an order of d. Right: Strong robust aggregator defenses, applying a single threshold based on variance to restrict corrupted mean to a constant distance.

training examples, misclassifying them to any class other than the correct one generically. Effectively, the ML model updates in each SGD step learn little useful information. The final trained model is unusable, leading to denial-of-service.

Untargeted poisoning attacks are a subclass of poisoning attacks. There are 2 other prominent categories: targeted [30] and backdoor [1] poisoning. Targeted attacks force the trained ML model to misclassify a specific class of inputs. Backdoor attacks misclassify inputs with artificially planted trigger patterns. We do not study these 2 categories here, though we believe our key ideas can be adapted to them.

Prior untargeted attacks assume two types of attacker knowledge about the uncorrupted samples X, i.e., full-knowledge and partial knowledge. In the full knowledge setting, the adversary has direct access to X and is able to arbitrarily manipulate an $\epsilon < 1/2$ fraction of inputs from X. In the partial-knowledge setting, the adversary does not have access to the complete X. In particular, for instance, it cannot exactly compute the mean of X since it only knows its own uncorrupted vectors, not those of other clients, as is expected in typical federated learning setups. The rest of the capabilities in the partial-knowledge setup are the same as in the full-knowledge setting. We experimentally evaluate both these settings. Our formal analysis, though, is restricted to the full-knowledge case because it is difficult to generically say how much the subset known to the adversary statistically deviates from the full X in different practical setups.

3. Robust Aggregators

Given vectors Y created from corrupting some $\epsilon \cdot n$ in X, the aggregator does not know which elements in Y are corrupted. One principled way is to solve the problem is to distinguish *inliers* from *outliers* in Y, and lower the contribution of the latter when computing a measure of centrality. Nearly all robust aggregators work with this principle, either explicitly or implicitly, but differ in how they do so.

Robust aggregators with weak bounds on bias. Weak robust aggregators, such as Trimmed Mean and Median,

Algorithm 1 Meta-algorithm for strong robust aggregators

Input ϵ-corrupted set $Y = \{y_1, ..., y_n\} \subseteq \mathbb{R}^d$, n, ϵ, and $||\Sigma||_2$
Output $\tilde{\mu}$ robust mean

$\xi := k \cdot ||\Sigma||_2$ ▷ Choose $\sqrt{20} < k \leq 9$ [14], [25]
$Y' = Y$
for $j = 0$ **to** $j = 2 \cdot n \cdot \epsilon - 1$ **do**
  if $||\text{Cov}(Y')||_2 \leq \xi$ **then**
    return $\tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} y'_i$
  else
    $Y' \leftarrow$ OutlierRemovalSubroutine(Y', ϵ, $||\Sigma||_2$)
  end if
end for
return $\tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} y'_i$

compute measures of centrality *per dimension*. When we compute central tendencies for each dimension separately, an adversary can always create a bias of $\Theta(\sqrt{\epsilon})$ in a dimension, proportional to the benign variance in that dimension, by corrupting ϵ fraction of X. This is true for X sampled from any distribution with bounded variance (see Sec. 3 in [25] and Fact 1.2 in [15]). So, it is difficult to bound the total bias better than $O(\sqrt{\epsilon d})$ by analyzing individual dimensions [25], [31].

Let us take the example of Trimmed mean for illustration. Fig. 1 (middle) shows a corrupted 2-d sample that induces high bias against it. The attacker can create $\epsilon \cdot n$ outliers positioned at the extreme of benign samples along each dimension, displacing the trimmed mean in each dimension by $\sqrt{\epsilon}$, giving total bias of $O(\sqrt{\epsilon d})$. Recent untargeted poisoning attacks have also experimentally shown that such high biases are practical on weak aggregators [2].

Robust aggregators with strong bounds on bias. Strong robust aggregators differ from weak ones in that they look at the magnitude of the vectors in Y along *all possible vector directions*, not just their individual components per dimension. They are able to provide substantially better upper bounds of $\tilde{O}(\sqrt{\epsilon})$ on bias, independent of d, because

Algorithm 2 FILTERING [17]

Input ϵ-corrupted set $Y = \{y_1, ..., y_n\} \subseteq \mathbb{R}^d$, $||\Sigma||_2$
Output $Y' = \{y'_1, ..., y'_n\}$ updated set Y removing or undermining outliers

1: $c_i := 1/n, \quad i \in [n]$ ▷ Initialize equal weights
2: $\mu_c := \sum_{i=1}^n c_i y_i$ ▷ Compute mean of Y
3: $v \leftarrow$ largest eigenvector of Cov(Y)
4: $\tau_i := \langle y_i - \mu_c, v \rangle^2, \quad i \in [|Y|]$
5: $c_i := c_i(1 - \frac{\tau_i}{\tau_{max}}), \quad i \in [|Y|]$
6: $cnt :=$ count vectors in Y with $c_i \neq 0$
7: $c_i := c_i / \sum_{i=1}^{cnt} c_i$ ▷ Normalize the new weights
8: **return** $\{c_1 \cdot y_1, \ldots, c_n \cdot y_n\}$

Algorithm 3 NO-REGRET [18]

Input ϵ-corrupted set $Y = \{y_1, ..., y_n\} \subseteq \mathbb{R}^d$, $||\Sigma||_2$
Output $Y' = \{y'_1, ..., y'_n\}$ updated set Y removing or undermining outliers

1: $l_{i,j} = ||y_i - y_j||_2$ for all $i, j \in [n]$
2: $\eta := 0.5/(\max(l_{i,j}))^2$ ▷ Step size for re-weighting
3: $c_i := 1/n, \quad i \in [n]$ ▷ Initialize equal weights
4: $\mu_c := \sum_{i=1}^n c_i y_i$ ▷ Compute mean of Y
5: $v \leftarrow$ largest eigenvector of Cov(Y)
6: $\tau_i := \langle y_i - \mu_c, v \rangle^2, \quad i \in [|Y|]$
7: $c_i := c_i(1 - \tau_i \cdot \frac{\epsilon\eta}{2||\Sigma||_2 d}), \quad i \in [|Y|]$
8: $\Delta_{n,\epsilon} := \{\tilde{C} | \sum \tilde{c}_i = 1, \tilde{c}_i \leq \frac{1}{(1-\epsilon)n}\}$
9: $C := \arg\min_{\tilde{C} \in \Delta_{n,\epsilon}} D_{KL}(\tilde{C}||C)$
10: **return** $\{c_1 \cdot y_1, \ldots, c_n \cdot y_n\}$

Algorithm 4 Sum of Squares (cf. [19])

Input ϵ-corrupted set $Y = \{y_1, ..., y_n\} \subseteq \mathbb{R}^d$, $||\Sigma||_2$
Output $Y' = \{y'_1, ..., y'_n\}$ updated set Y removing or undermining outliers

1: $W := \{w_1, \ldots, w_n\}$ ▷ create variables for weights
2: $Y' := \{y'_1, \ldots, y'_n\}$ ▷ create variables for Y'
3: $\mu' := \frac{1}{n} \sum_{i=1}^n y'_i$ ▷ create a variable for mean
4: $\Sigma' := \frac{1}{n} \sum_{i=1}^n (y'_i - \mu')^T \cdot (y'_i - \mu)$
5: $\tilde{\mathbb{E}} \leftarrow$ initialize a SoS program with W and Y'
6: $\tilde{\mathbb{E}}$.add($w_i^2 = w_i$ for every $i \in [n]$) ▷ add constraints
7: $\tilde{\mathbb{E}}$.add($\sum_{i=1}^n w_i = (1-\epsilon)n$)
8: $\tilde{\mathbb{E}}$.add($w_i y'_i = w_i y_i$ for every $i \in [n]$)
9: $\tilde{\mathbb{E}}$.add($\frac{1}{n} \sum \langle y'_i - \mu', v \rangle^2 \leq 9||\Sigma||_2$) for all $v \in \mathbb{R}^d$
10: $\tilde{\mathbb{E}}$.solve() ▷ solve the program to get operator $\tilde{\mathbb{E}}$
11: **return** $\{\tilde{\mathbb{E}}(y'_1), \ldots, \tilde{\mathbb{E}}(y'_n)\}$

Figure 2: Pseudocode for OutlierRemovalSubroutine in Alg. 1 as instatiated by 3 different strong robust aggregators.

of a key fact: If two sets share at least $(1-\epsilon)$ fraction of elements and exhibit equal maximum variance, then their respective means cannot deviate by more than $O(\sqrt{\epsilon})$ from each other (see the explanation below Fact 1.2 in [15]). The spectral norm $||\Sigma||_2$ of the covariance matrix captures the maximum variance across all vector directions (it is the largest eigenvalue). Using this principle, strong robust aggregators check if the variance of the corrupted inputs Y exceeds the expected $||\Sigma||_2$ known from clean samples.

The first polynomial-time strong aggregators have been proposed only in the last decade [15], [17]. Existing poisoning attacks fail to create much bias against them [14].

Alg. 1 is a common backbone for all polynomial-time strong aggregators. The basic idea is to compute a weighted average of vectors in Y. Lower weights are given to outliers. The way the weights are computed varies across different strong aggregators and is abstracted as the OutlierRemoval-Subroutine (Line 7). The entire procedure is iterative. In each iteration, the vectors in Y are re-weighted until the maximum variance of the newly weighted samples falls below a threshold ξ (Line 4). The threshold depends on $||\Sigma||_2$ (Line 1). This process ensures that all outliers in Y are assigned minimal weight, rendering the mean of the re-weighted points a reliable estimate of the true mean.

The value ξ depends on the natural variance ($||\Sigma||_2$) of the benign samples, trading off how many inliers vs. outliers the procedure is willing to exclude. If the defense chooses ξ much smaller than the natural variance of benign samples, it will filter out benign samples. For example, a constant fraction (about 0.27) of benign vectors are statistically expected to be between $||\Sigma||_2^{\frac{1}{2}}$ and $2 \cdot ||\Sigma||_2^{\frac{1}{2}}$ for normal (Gaussian) distributions—therefore, defenses which filter out points in this range are likely to have bias at least proportional to $\sqrt{\epsilon}$, for $\epsilon = 0.27$, even without any corruption. Therefore, practical defenses would use $\xi > k \cdot ||\Sigma||_2$, for some constant k. Prior works suggest using $k = \sqrt{20}$ [14] and $k = 9$ [17]. The provable bound on bias then is $\tilde{O}(\sqrt{\epsilon}) \cdot ||\Sigma||_2^{\frac{1}{2}}$ [17].

Fig. 1 (right) plots the behavior of Alg. 1 on the same example that creates a large bias against the weak trimmed mean. The samples outside the radius (dark solid circle) defined by ξ have near zero weight. The resulting estimate is much closer to the true mean than the trimmed mean.

The weight assignment procedure varies across all the different strong aggregators proposed in prior work: Filtering [17], No-Regret [18], and SoS [19]. Fig. 2 shows the different strategies to implement the OutlierRemovalSubroutine the 3 aggregators use—all follow the same meta-procedure of Alg. 1. Readers need not understand their details, they are included to make the paper self-contained. We experimentally evaluate them in Section 5, when possible.

Computational bottleneck. The maximum variance direction of Y (Line 3 in Alg. 2) is computationally expensive to compute. We will later show in Section 4.4 that it is fundamental to the problem of robust aggregation, not just to the above solutions. This computational bottleneck forces practical realizations of Alg. 1 to operate on smaller subsets of dimensions at a time, as explained in Section 4.

4. The HIDRA Attack

We propose a principled untargeted poisoning attack to defeat existing polynomial-time strong aggregators.

4.1. Warm up: Attack in Low Dimensions

The invariant in strong robust aggregators (see Alg. 1) is that Y', a re-weighted (scaled) version of Y, has maximum variance below a threshold ξ. The attack preserves this invariant: Given uncorrupted vectors X, it replaces $n \cdot \epsilon$ of them to construct Y' such that Cov(Y') is below the threshold. Thus, all the corrupted vectors will be used in the final weighted mean statistic computed by Alg. 1. The attack procedure is given in Alg. 5. It computes the mean vector

Algorithm 5 Pseudo-code describing HIDRA

Input benign set $X = \{x_1, ..., x_n\}$, fraction of corruption ϵ, variance threshold ξ
Output ϵ-corrupted set $Y = \{y_1, ..., y_n\}$

1: $\hat{\mu} := \frac{1}{n}\sum_{i=1}^{n} x_i$ ▷ compute the mean of X
2: $\hat{\Sigma} := \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^T \cdot (x_i - \hat{\mu})$
3: $\hat{s} := \frac{\hat{\mu}}{||\hat{\mu}||_2}$
4: $\sigma^2_{max} := \frac{\xi}{\sqrt{20}}$ ▷ estimate σ^2_{max} using ξ
5: $z = \sqrt{\frac{\xi - \sigma^2_{max}}{\epsilon^2 + \epsilon(1-\epsilon)^2}}$ ▷ magnitude of corruption along $\hat{s}$
6: $y_i := \hat{\mu} - \hat{s} \cdot z$ for all $i \in [1, n \cdot \epsilon]$
7: $y_i := x_i$ for all $i \in [n \cdot \epsilon + 1, n]$
8: **return** Y

$\hat{\mu}$ of X (line 1) and creates corrupted vectors that hit the maximum allowable deviation from the mean towards zero, without getting filtered out. Specifically, line 5 computes the exact magnitude of the corruptions to create along the direction of $-\hat{\mu}$ such that after corrupting $n \cdot \epsilon$ vectors (as in line 6), the resulting Y has maximum variance below ξ.

To achieve that, recall that the threshold ξ for which the guarantees of the robust aggregator hold, is always higher than the benign sample variance along any direction. Ideally, the threshold should be $\geq 9 \cdot \sigma^2_{max}$ as given by theoretical analyses in prior works [17]. Prior implementations for FILTERING and NO-REGRET [14] use $\sqrt{20}$ times a constant estimate of σ^2_{max} as the threshold. We make a key observation that there exists a direction $\hat{s}$, along which corrupted vectors can shift the aggregate far from the original mean to the maximum value, while ensuring that the variance in all other directions remains below the threshold. Specifically, using the magnitude z given in Eqn. (2) below increases the bias as much as possible without exceeding the threshold. We use the offset $-z\hat{s}$ for all corrupted vectors.

$$z = \sqrt{\frac{\xi - \sigma^2_{max}}{\epsilon^2 + \epsilon \cdot (1-\epsilon)^2}} - \mu^{\hat{s}} \tag{2}$$

$$\mu^{\hat{s}} = \frac{1}{n}\sum_{i=0}^{n} \langle x_i, \hat{s} \rangle \tag{3}$$

Our analysis shows that the above strategy will not increase variance in any other direction beyond the threshold (see Lemma 1.2 in Section 4.3). This is necessary to not trigger the outlier removal procedure. The magnitude of corruption is thus maximized, subject to the threshold invariant remaining valid. Our analysis presented in Section 4.3 proves that using the value of z given in Eq 2, HIDRA hits the theoretical upper bound on the bias. We confirm it experimentally for practical setups as well in Section 5. Fig. 3 illustrates the corruption strategy described above, depicting the placement of all corrupted vectors just inside the variance threshold boundary.

The direction of the perturbation, i.e., along $-\hat{\mu}$, is chosen to reduce the final $\hat{\mu}$ closer to zero, motivated specifically in untargeted poisoning attacks. If the updates in SGD are close to zero, they carry lesser signals about the training samples, effectively disallowing the model from learning. Other forms of poisoning can use other directions based on their goals, but we focus only on untargeted poisoning.

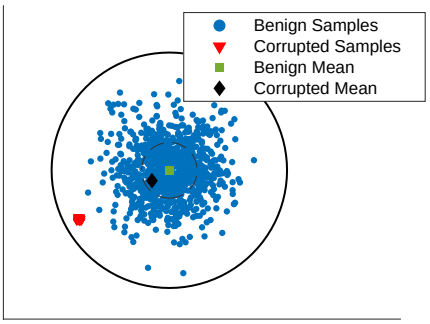

Figure 3: HIDRA : Corruptions crafted within variance threshold, yet biasing the mean to an order of $\Omega(\sqrt{\epsilon})$

Note that our attack does *not* have bias dependent on d in low dimensions, as expected from prior theoretical analysis of strong robust aggregators. Our attack does close the gap between the best-known attacks and the upper bound, showing the tightness of prior theoretical analyses.

4.2. Vulnerability in High Dimensions

The analysis of strong robust aggregators makes idealized computational assumptions. The expensive steps in these defenses are computing the maximum variance and a corresponding maximum variance direction of inputs Y. Recall that the maximum variance corresponds to the spectral norm of the sample covariance matrix $||\text{Cov}(Y)||_2$, i.e., the largest eigenvalue. The maximum variance direction is the direction of one of the eigenvectors with largest eigenvalue. The exact computation of $||\text{Cov}(Y)||_2$ is $O(d^3)$ [21] due to iterative matrix multiplication steps. As a rough estimate, it takes about 150 seconds on a single CPU core of a modern desktop to compute the largest eigenvector sequentially and its eigenvalue for d=10^4. Averaging in SGD is over gradient vectors which can be much larger, for instance, small CNNs need $d = 3 \times 10^5$ and modern large language models can have $d = 10^{10}$ to 10^{12}. Therefore, given finite memory per computational device, the computation over $d \times d$ matrices has to be split into smaller matrices when d is large.

Practical realization of strong robust aggregators. To overcome the computational bottleneck, recent works split the dimensions into disjoint chunks that are small enough to be computed on individually [13], [14]. We illustrate how practical realization of strong robust aggregators do this. Let the original dimension of samples be d and FILTERING be the robust aggregator chosen. Then, we take the first m dimensions of all vectors as the first chunk, the next m dimensions of all vectors as the second chunk, and so on. Each vector is partitioned into chunks of size say $m = 1000$. This will result in each vector having $c = \lfloor \frac{d}{m} \rfloor$ chunks.

Algorithm 6 Practical realizations of strong robust aggregators in high dimensions

Input ϵ-corrupted set $Y = \{y_1, \ldots, y_n\} \subseteq \mathbb{R}^d$, partition size m, $||\Sigma||_2$, a strong aggregator $\mathcal{A}$ following Alg. 1
Output Robust mean of Y, $\mu \in \mathbb{R}^d$

1: $\mu = [0, \ldots, 0]$ ▷ initialize with zero vector in $\mathbb{R}^d$
2: $i = 0$
3: **while** $i < d$ **do**
4: $\tilde{Y} \leftarrow$ set of sampled Y using indices in $[i, i+m-1]$
5: $\mu[i, i+m-1] := \mathcal{A}(\tilde{Y}, ||\Sigma||_2)$ ▷ get chunk aggregate
6: $i := i + m$
7: **end while**
8: **return** μ

The FILTERING aggregator will be run on the first chunk, i.e., first m dimensions of all samples to find the robust aggregate, and similarly on each subsequent chunk. We get c robust aggregates, one for each chunk, which are then concatenated to output the final output vector of dimension d. This generic chunking strategy is detailed in Algorithm 6.

Vulnerability. Chunking introduces a new source of vulnerability. Running robust aggregation on different chunks independently creates an opportunity for the adversary to bias the result on each chunk separately. Specifically, the adversary can bias the aggregate of each chunk by $\Omega(\sqrt{\epsilon})$ using Algorithm 5. Since the final aggregated vector is the concatenation of all of the aggregates of each chunk, the biases from the aggregates from all chunks add up. Therefore, the bias in the final aggregated model is $\Omega(\sqrt{\epsilon c})$.

Since the number of chunks c is proportional to d, the total bias introduced by HIDRA is $\Omega(\sqrt{\epsilon d}) \cdot \sqrt{\xi}$. Formally, Theorem 1 in Section 4.3 proves our claim.

Multi-chunk HIDRA. Multi-chunk strong robust aggregators have to decide the threshold to use per chunk. Prior work has used a constant function to bound the variance of each chunk by a fixed constant σ^2_{max}, which is the maximum value of the variance across all chunks (line 5) [14]. Our attack therefore uses σ^2_{max} as a replacement for $||\Sigma||_2$ in Alg. 5. One can extend the defense to consider an adaptive threshold, a different constant multiple of the variance σ_i^2 for chunk i. The HIDRA attack would simply use Alg. 5 for each chunk separately using the respective threshold. The analysis remains the same because the threshold used in each chunk must be proportional to the natural variance σ_i^2 in that chunk, in order to avoid filtering out inliers. So, the threshold used must be at least a constant times σ_i^2.

Full vs. Partial Knowledge Setups. Algorithm 5 extends to the partial-knowledge setting as well. The only change is to estimate the $\hat{\mu}$ (see Line 1) using a subset of benign vectors that the adversary has access to (e.g. its own). We have evaluated both setups in Section 5.

In summary, we have shown HIDRA, an attack strategy to create a bias of $\Omega(\sqrt{\epsilon d})$ in high dimensions. This defeats the main advantage offered by strong robust aggregators over weak ones. Practical incarnations of strong robust aggregators that give $\tilde{O}(\sqrt{\epsilon})$ bias, therefore, remain elusive.

4.3. Theoretical Optimality Analysis

Our Theorem 1 asserts that HIDRA against practical realizations of strong robust aggregators will result in a bias that will be at least $\Omega(\sqrt{\epsilon d})$. Prior known bias for such aggregators is $\tilde{O}(\sqrt{\epsilon d})$, so our attack is nearly optimal. Our analysis presented here is for the full-knowledge setting. It remains the same for the partial-knowledge setting, modulo the estimation error in computing $\hat{\mu}$ (Line 1 in Alg. 5), which varies by datasets.

Theorem 1. HIDRA *, as outlined in Algorithm 5, will result in a bias of* $\Omega(\sqrt{\epsilon d}) \cdot ||\Sigma||_2^{\frac{1}{2}}$ *against Alg. 6 in the worst case.*

Proof. We prove three Lemmas 1.1, 1.2, and 1.3 that are stated below to prove this theorem. We provide complete proofs for the Lemmas in Appendix A.

In the Lemma 1.1, we first prove that along any direction $\hat{s}$, we can corrupt an ϵ fraction of samples to increase the variance along that direction to the maximum possible value defined by the threshold ξ. We show it suffices to use a magnitude of perturbation close to that given in Eq. 2 previously, to create $\Omega(\sqrt{\epsilon}) \cdot ||\Sigma||_2^{\frac{1}{2}}$ bias with high probability.

In the second Lemma 1.2, we claim that if the corrupted vectors have magnitude of $z = \sqrt{\frac{\xi - \sigma^2_{max}}{\epsilon^2 + \epsilon(1-\epsilon)^2}} - \mu^{\hat{s}}$ the maximum variance will be ξ and the maximum variance direction will be $\hat{s}$. Thus, none of the points will be filtered out (or reweighted) by strong robust aggregators (see Alg. 1).

It follows from Lemma 1.1 and 1.2 that HIDRA designs corruptions that do not get filtered out by strong robust aggregators and create $\Omega(\sqrt{\epsilon}) \cdot ||\Sigma||_2^{\frac{1}{2}}$ bias per chunk.

Finally, Lemma 1.3 proves that if the robust aggregator runs independently over c different chunks of original vectors, as done in Alg. 6, with threshold ξ, then the bias is $\Omega(\sqrt{\epsilon c}) \cdot \sqrt{\xi}$. When $\sqrt{\xi} \geq ||\Sigma||_2^{\frac{1}{2}}$ and $c = \lfloor \frac{d}{m} \rfloor$ where m is a constant representing the size of each chunk such that $m \ll d$, then the bias is about $\Omega(\sqrt{\epsilon d}) \cdot ||\Sigma||_2^{\frac{1}{2}}$. □

Lemma 1.1. *Let $\hat{s}$ be any direction, $\mu^{\hat{s}}$ be the component of benign mean along $\hat{s}$, and $\sigma_{\hat{s}}$ be the variance along $\hat{s}$. Then, replacing an $\epsilon \cdot n$ vectors in X with $-z(\hat{s})$, where $z = \sqrt{\frac{\xi - \sigma^2_{\hat{s}}}{\epsilon^2 + \epsilon \cdot (1-\epsilon)^2}} - \mu^{\hat{s}} \pm \frac{\delta \cdot \sigma_{\hat{s}}}{\sqrt{n}}$, results in bias of $\Omega(\sqrt{\epsilon})||\Sigma||^{\frac{1}{2}}$ with probability at least $1 - \exp(-\delta^2)$, for all $\delta > 1$.*

Lemma 1.2. *Let $\hat{s}$ be in the direction of the benign aggregate mean $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ and Y be ϵ-corrupted set of vectors. Let the corrupted vectors be along $-\hat{s}$ with magnitude $z = \sqrt{\frac{\xi - \sigma^2_{max}}{\epsilon^2 + \epsilon(1-\epsilon)^2}} - \mu^{\hat{s}}$. Then, $||\Sigma_Y||_2 \leq \xi$, where $||\Sigma_Y||_2$ is the spectral norm of the covariance matrix of Y.*

Lemma 1.3. *If strong robust aggregators are used to robustly aggregate over c chunks, as in Algorithm 6 with a single threshold ξ, then* HIDRA *performed over each chunk, as in Algorithm 5, will result in a bias of $\Omega(\sqrt{\epsilon c}) \cdot \sqrt{\xi}$.*

Elements of Construction (EC)

1) $\mathcal{D} \leftarrow d$ dimensional spherical Gaussian distribution (O, I)
2) $Y = \{y_1, \ldots, y_{n-n\epsilon}\} \overset{\text{i.i.d.}}{\sim} \mathcal{D}$:
   a) $\frac{1}{n}\sum_{i=1}^{n-n\epsilon} y_i = \hat{\mu}$
   b) $\text{Cov}(Y) = \hat{I}$ where $\hat{\mu} \approx O$ and $\hat{I} \approx I$
3) $B = \{b_1, b_2\} \leftarrow$ randomly chosen 2 orthogonal unit vectors
4) $L_i = \{\sqrt{d} + l_{i1}, \ldots, \sqrt{d} + l_{i\frac{n\epsilon}{2}}\} \leftarrow$ distance of corrupted vectors from the origin along ith vector in B such that :
   a) $|l_{ij}| \ll \sqrt{d} \quad \forall j \in [1, \frac{n\epsilon}{2}]$ for instance, $|l_{ij}| \in [0, 1]$
   b) $\sum_{j=1}^{\frac{n\epsilon}{2}} l_{ij} = 0$
   c) $\sum_{j=1}^{\frac{n\epsilon}{2}} l_{1j}^2 = \sum_{j=1}^{\frac{n\epsilon}{\kappa}} l_{2j}^2$
5) $c_1, c_2 \leftarrow$ set of corrupted vectors along each vector in B such that for each c_i: $c_i = \{(\sqrt{d} + l_{i1}) \cdot b_i, \ldots, (\sqrt{d} + l_{i\frac{n\epsilon}{2}}) \cdot b_i\}$
6) $C = \{y_{n-n\epsilon+1}, \ldots, y_n\} \leftarrow c_1 \cup c_2$
7) $Y' = Y \cup C$

4.4. Is Computational Bottleneck Fundamental?

HIDRA works against several strong robust aggregators, all of which face the computational bottleneck of computing the maximum variance direction of given vectors. It corresponds to computing the largest eigenvector, a problem of broad interest in ML that does not scale with dimensions. How fundamental is the connection between designing a strong robust aggregator and computing the maximum variance direction of the given vectors? Specifically, *is it necessary for strong robust aggregators to be as computationally expensive as finding the maximum variance direction?*

Bias vs. computational complexity. We will construct a set of vectors, call it Y', for which computing its maximum variance direction approximately will have complexity not much worse than that of computing the aggregate with a strong robust aggregator f. Theorem 2 given later formalizes the claim. To the best of our knowledge, the worst-case time complexity of computing the maximum variance direction approximately[6] for the set Y' is $min(\tilde{O}(n^2 d), O(d^3))$ [21], [22], [32] and hence, devising more efficient f would imply faster algorithms for the former on Y'. We refer to such sets Y' as a *non-trivial instances* for computing the maximum variance direction. To create Y' we rely on an initial set of vectors sampled from a spherical Gaussian [7] distribution as it suffices to show the existence of such sets for which the reduction is valid.

Constructing Y'. We follow the aforementioned elements of construction to construct the set Y'. Consider a d dimensional spherical Gaussian distribution with mean at Origin and I as covariance (1). Now, sample a set of vectors Y, i.i.d from this distribution as shown in (2). We construct the set Y' from Y by adding $n\epsilon$ *corrupted vectors* to Y following the steps from (3) to (7). First, we choose a set B, as per (3), which represents 2 mutually perpendicular directions. Now we distribute $n\epsilon$ vectors equally along these 2 directions. The distances of these vectors from the origin in each of the directions in B are given in (4). Informally, all vectors are slightly different from each other but are close to $\sqrt{d}$ distance from the origin. Accordingly, we get $\frac{n\epsilon}{2}$ corrupted vectors in each of these directions as given in (5). Finally, we add these vectors to the set Y to create Y' (6, 7).

Note that while constructing Y' we choose the corrupted vectors to be at distances of about $\sqrt{d}$ to make them indistinguishable from the benign vectors of Y, since samples from a d dimensional Gaussian are at a distance of $\sqrt{d}$ with very high probability [33] (also see Figure 2.1 in [33] for illustration). Further, the different added L_is make sure that the corrupted vectors are also different from each other with respect to their magnitude (L_2 norm). Hence, the corrupted and benign vector magnitudes follow the same distribution.

We first establish in Lemma 2.1 that the maximum variance of vectors in Y' lies approximately along the difference of vectors in B. We provide the proof in Appendix A.

Lemma 2.1. *Given an ϵ-corrupted set of vectors Y', as constructed using the elements of construction (EC), the direction of maximum variance of Y' lies with a small angle of $cos^{-1}\left(\sqrt{1 - O(\frac{\delta^2}{n})}\right)$ from the difference of vectors in set B with probability at least $1 - 2\exp(\frac{-\delta^2}{2})$, for all $\delta > 2$.*

We see that the approximation error in the angle drops with n. The failure probability drops exponentially in δ, which is a tunable constant in the analysis.

Reduction. Lemma 2.1 provides a way to compute the maximum variance direction of Y' if we know the difference of vectors in B. So, how does one find this difference given Y'? We provide a reduction in Alg. 7 to show that the direction of maximum variance of any set Y', constructed as above, can be computed approximately using the outputs of robust aggregator $f(Y')$ in quasilinear extra time $\tilde{\Theta}(|Y'|)$. Theorem 2 presented later formalizes the precise claim.

Alg. 7 works as follows. It computes two vectors μ', the average of Y' (line 1) and $\hat{\mu}$, the robust aggregate $f(Y')$ (line 2). Then, it iterates over all the vectors in Y' and checks for the projection of $\mu' - \hat{\mu}$ along each of them

6. We seek approximations where the approximation error decreases linearly with the number of vectors n. With regards to such error, approximate methods like power iteration operate with a time complexity of $\tilde{O}(n^2 d)$ [32].

7. These are distributions with equal variance along every direction.

Algorithm 7 Reduction Algorithm

Input $Y' = \{y_1, \ldots, y_n\} \subseteq \mathbb{R}^d$, strongly-bounded robust aggregator algorithm f
Output v^*, direction of maximum variance

1: $\mu' := \frac{1}{n}\sum_{i=1}^{n} y_i$ ▷ compute average of the set
2: $\tilde{\mu} := f(Y')$ ▷ compute robust mean of the set
3: $S := \{\}$ ▷ create an empty set
4: **for** $k = 1$ **to** $k = n$ **do**
5: $s_i := \langle \mu' - \tilde{\mu}, \frac{y_i}{||y_i||_2} \rangle$
6: $S \leftarrow s_i$ ▷ add s_i to S
7: **end for**
8: $C' \leftarrow$ set of $n\epsilon$ vectors with highest value in S
9: $c_1 := C'[1]$ ▷ take the first vector of C'
10: **for** $k = 2$ **to** $k = n\epsilon$ **do**
11: $c_2 := C'[k]$
12: **if** $\langle c_1, c_2 \rangle = 0$ **then**
13: **return** $\frac{c_1}{||c_1||_2} - \frac{c_2}{||c_2||_2}$
14: **end if**
15: **end for**
16: **return** $\frac{c_1}{||c_1||_2} - \frac{c_2}{||c_2||_2}$

(lines 4-6). The $n\epsilon$ vectors with the largest projections are identified as the set of corrupted vectors C' (line 8). Next, it iterates over C' to find a pair of mutually perpendicular vectors. Finally, it takes the unit vectors along each of the mutually perpendicular vectors and returns the difference of these unit vectors as output (lines 9-16).

Lemma 2.2 below states that Alg. 7 indeed finds the difference of vectors in set B with very high probability.

Lemma 2.2. *Let $f(Y') = \tilde{\mu}$ and $\mu' = \frac{1}{n}\sum_{i=0}^{n} Y'[i]$, then Alg. 7 returns the difference of vectors in set B with probability at least $1 - n \cdot \exp(\frac{-n\delta^2}{2})$, for all $\delta > 1$.*

Proof Sketch. We defer the full proof to Appendix A and provide a sketch here. We show that the projection of $\mu' - \tilde{\mu}$ is much larger along the corrupted vectors than along others in set Y' with very high probability. Hence, it is sufficient to iterate over vectors in Y' and get the top $n \cdot \epsilon$ vectors in the order of $\mu' - \tilde{\mu}$'s projections along them to find the corrupted ones. Thus, in Line 8, C' represents the set of corrupted vectors. Since equal numbers of vectors in C' align along each vector in set B, if we choose any vector from C' we can find a perpendicular corrupted to it in a single iteration. The difference of these corrupted vectors normalized by their norm is equal to the difference of vectors in set B. It implies Alg. 7 returns the difference of vectors in set B. □

Using Lemmas 2.1 and 2.2, we state and prove the desired Theorem 2 below. We use the $\tilde{\Theta}$ version of the asymptotic complexity in the proof, which ignores logarithmic factors.

Theorem 2. *Let f be a strong robust aggregator with running time $T(f)$ and Y' be a set of vectors constructed as in EC. Then, the direction of maximum variance of Y' can be computed with a small approximation error, in time $T(f) + \tilde{\Theta}(|Y'|)$, with very high probability.*

Proof. Given the set Y', Lemma 2.1 asserts that the direction of maximum variance of Y' lies approximately along the difference of vectors in set B. The approximation error in the angle is stated in Lemma 2.1 and approaches zero with increasing number of samples n. Lemma 2.2 asserts that the reduction stated in Algorithm 7 finds the difference of vectors in set B using Y'. The total failure probability in these two Lemmas is either exponentially small in n or in δ, so taking a union bound, the final result is correct with very high probability. The time taken by Algorithm 7, excluding Line 2, is $\tilde{\Theta}(n \cdot d)$, which can be verified by inspection. Each vector summation and dot product $\langle \cdot \rangle$ is O(d). The 2 loops run $O(n)$ times, with each iteration O(d). Picking the highest $n\epsilon$ values in S (Line 8) takes $\tilde{\Theta}(n \cdot d)$, which is the same as $\tilde{\Theta}(|Y'|)$. The Line 2 takes time $T(f)$. So, the total running time of Alg. 7 is thus $T(f) + \tilde{\Theta}(|Y'|)$. □

We have shown that the existence of efficient strong aggregators f implies being able to compute the approximate direction of maximum variance, for non-trivial inputs as in EC. The latter problem, to the best of our knowledge however, has $O(d^3)$ algorithms for general inputs. We are not aware of any faster solutions for the input class EC.

5. Evaluation

Goals. Our experimental evaluation of machine learning tasks aims to answer two primary questions:

1) Does the bias introduced by HIDRA during training match the theoretical bias our analysis expects?
2) How effective is HIDRA as an untargeted poisoning attack, i.e., what drop in model accuracy does it induce when used at each training step?

We evaluate on standard image classification datasets, training with SGD using state-of-the-art strong robust aggregators with strong bias bounds. We evaluate both the partial and full knowledge settings. Furthermore, we aim to compare the impact of our attack on training accuracy with that of existing attacks in these scenarios.

5.1. Experimental Setup

Untargeted poisoning attacks [2], [3] are most often evaluated in federated learning setups, which consist of several untrusted clients and a trusted server. Clients train locally and send local model updates (vector) which are aggregated by the server as discussed in Section 2. We evaluate in this setup assuming an ϵ fraction of the clients are malicious and send update vectors created using HIDRA.

Datasets. We use 3 image classification datasets: MNIST [34], Fashion-MNIST [35], and CIFAR10 [36]. Each of these datasets comprises $60,000$ training and $10,000$ test examples split across 10 classes. In MNIST and Fashion-MNIST, the examples are 28×28 grayscale images, while in CIFAR10, they are 32×32 color (RGB) images.

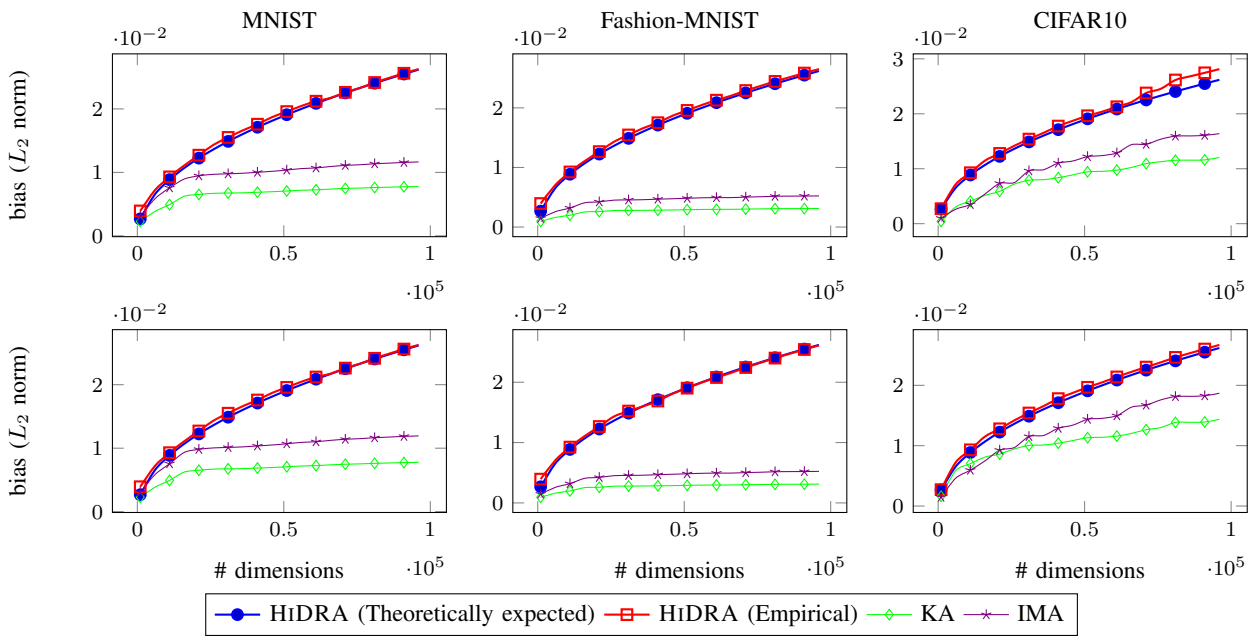

Figure 4: Bias vs. # of Dimensions against FILTERING (top) and NO-REGRET (bottom) strong aggregators.

Models and training. We use convolutional neural networks (CNN) with two convolutional layers followed by two fully connected layers with ReLU activations. The CNNs used for MNIST, Fashion-MNIST, and CIFAR10 have 3.2×10^5, 3.2×10^5, and 1.7×10^6 parameters respectively, which are the dimensions d of the vectors being aggregated. In all experiments, the data is independently and identically distributed across 100 federated clients. We fix $\epsilon = 0.2$ for all of our experiments to match the setup with the prior work that proposes strong robust aggregators [14]. We also report on varying ϵ at the end of this section. In each round of federated learning, every client runs 5 local epochs with batch size 10 before submitting the updated local model to the server. We fix the learning rate at 0.001 for MNIST and Fashion-MNIST, training the models for 100 rounds. In the case of CIFAR10, we use a learning rate of 0.01 and extend the training to 400 rounds. We design and implement all our experiments using Python and PyTorch. All the experiments are conducted on Ubuntu 20.04 LTS servers with 64 AMD Ryzen Threadripper 3970X CPUs, 96G RAM, and 2 NVIDIA GeForce RTX 3090. The code for evaluation is provided in [37].

Strong robust aggregators. We evaluate all strong robust aggregators that are tractable to run on high-dimensional vectors. These include FILTERING and NO-REGRET. We follow the setup identical to prior work, where the chunk size is 1000, the threshold ξ per chunk is $\sqrt{20} \times 10^{-5}$, and the estimated upper bound for σ^2_{max} is 10^{-5} for each chunk [14]. We cannot run the third known strong robust aggregator, called SoS, since it requires solving a system of degree-4 equations using a semi-definite programming (SDP) solver. Current SDP solvers cannot do so in a reasonable time for $d > 10$, so it is intractable for our tasks.

Prior attacks. There are 3 untargeted poisoning attacks considered in prior work evaluating strong aggregators [14]: Krum Attack (KA) [2], Trimmed Mean Attack (TMA) [2] and Inner-Product Manipulation Attack (IMA) [3]. We report the bias, and drop in model accuracy, achieved by KA and IMA against all tractable defenses we evaluate.

We evaluated TMA for 20 training iterations on all defenses on all datasets, but we do not report on TMA in detail here. This is because all strong aggregators run very effectively against TMA, but take too much time to run in full. Specifically, we verified that the defenses filter out (assign zero weight to) *all the poisoned gradients* created by TMA—unlike for other attacks—in all iterations we tested, so the bias induced by TMA is zero. But such filtering by defenses is iterative and removes each of the $n \cdot \epsilon$ poisoned gradients one at a time, taking $O(d^3)$ time for each. This makes the training very slow (and pointless) to run in full.

5.2. Empirical vs. Theoretical Bias

In Section 4.3, we prove that HIDRA will result in a bias of at least $\sqrt{\epsilon c}\sqrt{\frac{\xi}{\epsilon+(1-\epsilon)^2}}$. Since the chunk size is 1000, the theoretically anticipated bias is $\frac{\sqrt{\epsilon d}}{\sqrt{1000}} \cdot \sqrt{\frac{\xi}{\epsilon+(1-\epsilon)^2}}$.

We plot the empirical and the theoretical bias using HIDRA on all the 3 datasets against FILTERING and NO-REGRET defenses in Fig. 4. Specifically, for each configuration, consisting of a dataset and a defense, we plot the bias introduced by HIDRA for increasing number of dimensions. The empirical bias created by prior attacks KA and IMA attacks is also shown. We choose an arbitrary training round

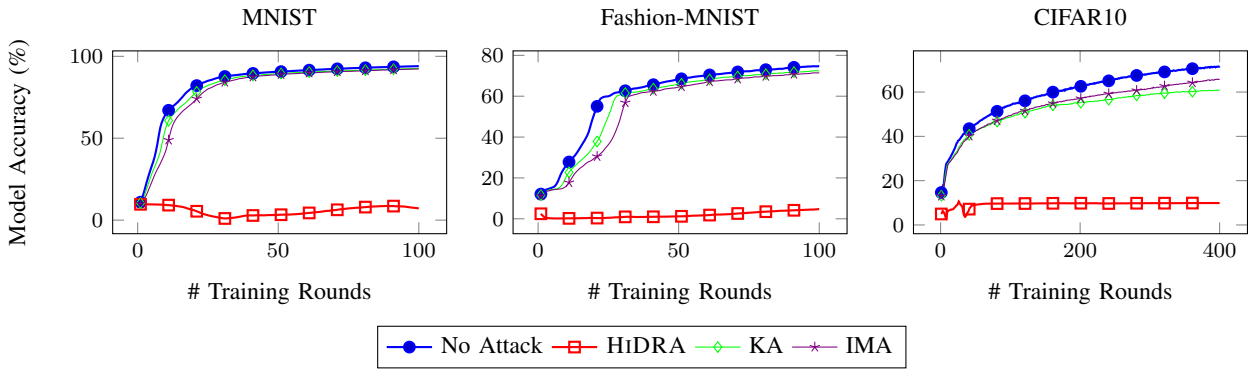

Figure 5: Impact of HIDRA on accuracy against FILTERING

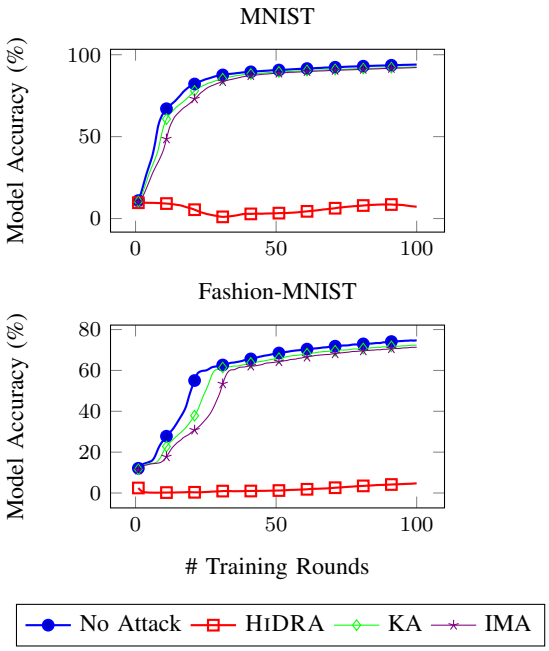

Figure 6: Impact of HIDRA on accuracy against NO-REGRET

number 10 to plot the bias, however, the observations are largely the same for all training rounds we sampled.

The empirical efficacy of HIDRA aligns almost exactly with the theoretical bias anticipated. This shows that our analysis is tight. Our result also shows that when HIDRA is used against the state-of-the-art strong robust aggregators the bias increases proportionately to $\sqrt{d}$, confirming our main claim. Further, observe that the bias introduced by other attacks is much smaller than HIDRA, highlighting the drastic efficacy gains achieved. The efficacy of prior attacks also increases marginally as d, since they introduce some bias in each chunk that adds up across all chunks. But the difference in efficiency between HIDRA and prior attacks widens sharply as d increases.

HIDRA induces bias proportional to $\sqrt{\epsilon d}$ against strong robust aggregators, showing a contrast to the idealized analysis of these aggregators. Prior attacks have efficacy well below that of our attack.

5.3. Impact of HIDRA on Accuracy

The main motivation for untargeted poisoning attacks is to reduce the overall performance of the trained model by introducing bias. Therefore, we also measure how the classification accuracy of the models gets affected after end-to-end training. We find that HIDRA has a sharp and negative effect on the model performance, unlike prior attacks which do not affect the model performance much at all when trained with strong robust aggregators.

First, we report the results for the full-knowledge setting and later report the results for the partial-knowledge setting.

HIDRA on FILTERING. Fig. 5 captures the performance of all the considered attacks against FILTERING. HIDRA causes the accuracy of CNN on the MNIST dataset to drop by $\mathbf{87}\%$ (from $\mathbf{94}$ to $\mathbf{7}\%$). Similarly, on the Fashion-MNIST dataset and CIFAR10 datasets, the accuracy drops by $\mathbf{70}\%$ and $\mathbf{62}\%$. Existing attacks do not affect the accuracy much. We verify that this is because corrupted vectors created by prior attacks are filtered out by strong robust aggregator defenses. In contrast, none of the added corrupted vectors added by HIDRA are filtered out.

HIDRA on NO-REGRET. Fig. 6 illustrates the impact of HIDRA on the model performance against NO-REGRET. HIDRA drastically reduces accuracy by $\mathbf{87}\%$ and $\mathbf{70}\%$ for MNIST and Fashion-MNIST respectively. NO-REGRET is computationally d times more expensive than FILTERING, hence, it is not practical to evaluate it on the CIFAR10 dataset which has at least $5\times$ higher d than the other one. It would take several days to train one model using NO-REGRET on CIFAR10 using our testbed. Therefore, we report results on MNIST and Fashion-MNIST only.

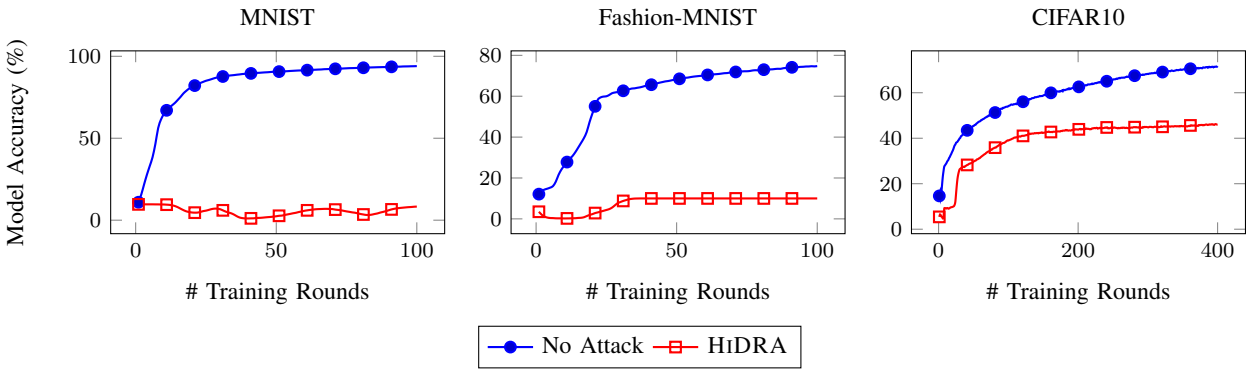

Figure 7: Impact of HIDRA (Partial Knowledge) on accuracy against FILTERING.

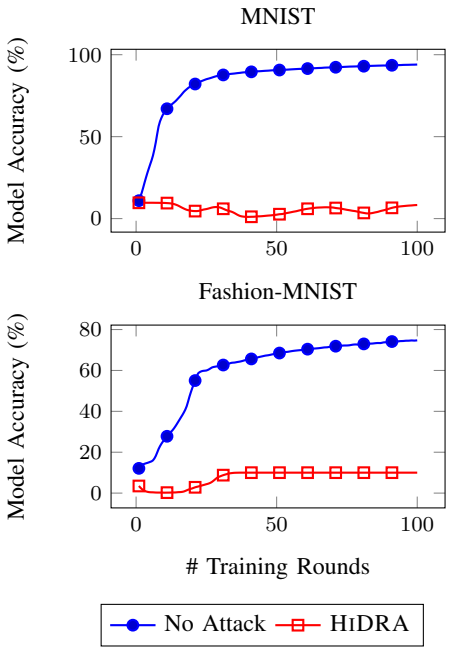

Figure 8: Impact of HIDRA (Partial-Knowledge) on accuracy against NO-REGRET.

HIDRA completely destroys the ML model performance for all datasets against robust aggregator defenses we evaluated in the full-knowledge setting.

Partial-knowledge setting. We consider only the gradients of the first $\epsilon \cdot n$ samples chosen randomly in each SGD round are seen and controlled by the adversary. HIDRA works effectively in this setting as well as shown in Fig. 7. HIDRA reduces the model accuracy by $\mathbf{86}\%$, $\mathbf{65}\%$, and $\mathbf{26}\%$ in MNIST, Fashion-MNIST, and CIFAR10 datasets respectively against FILTERING. The results are similar against NO-REGRET (see Fig 8). The impact of HIDRA in the partial-knowledge setting is much more pronounced than that of prior attacks even with full-knowledge.

HIDRA with partial-knowledge is more effective even when compared to prior attacks with full-knowledge.

One expects the attack efficacy to reduce in the partial-knowledge setting compared to the full-knowledge. This is because the adversary has to estimate the direction of the benign mean using only points it can see, i.e., its own vectors. Our result shows that the difference between the two settings, though visible, is relatively small on real datasets.

HIDRA performance with varying ϵ. We have reported all the results at $\epsilon = 0.2$ so far. We show the impact of HIDRA for varying $\epsilon = \{0.01, 0.05, 0.1, 0.2\}$, the fraction of corrupted samples, in the full-knowledge setting.

Fig. 9 shows the results. HIDRA continues to exhibit a large drop in model accuracy even at lower fractions. For instance, at $\epsilon = 0.05$, HIDRA lowers the performance by $\mathbf{74}\%$ on Fashion-MNIST. At $\epsilon = 0.1$, the drop induced is over 20% for CIFAR10. The sharpest drops occur at $0.1 \leq \epsilon < 0.2$ for MNIST and CIFAR10.

HIDRA can induce over 70% loss of model accuracy even at smaller $\epsilon = 0.05$ on some datasets. The sharpest drop in accuracy is between ϵ of 0.1 and 0.2 for others.

6. Related Work

Byzantine Robust aggregation has its roots in robust statistics, a subject with a rich history [24], [38]. The threat of poisoning attacks on large neural nets has brought urgent attention to it, and in particular, to the challenge of minimizing bias when aggregating high-dimensional vectors.

Poisoning attacks. Poisoning attacks are a long-standing issue for ML in adversarial environments [39]. Their threat was highlighted early in federated learning systems for general ML classifiers [40] and neural networks [29]. Since then there has been an evolving cat-and-mouse game between poisoning attacks [1], [2], [13], [41], [42] and defenses to mitigate them [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53]. Targeted poisoning attacks aim to train models that misclassify a certain class of inputs [30], [54].

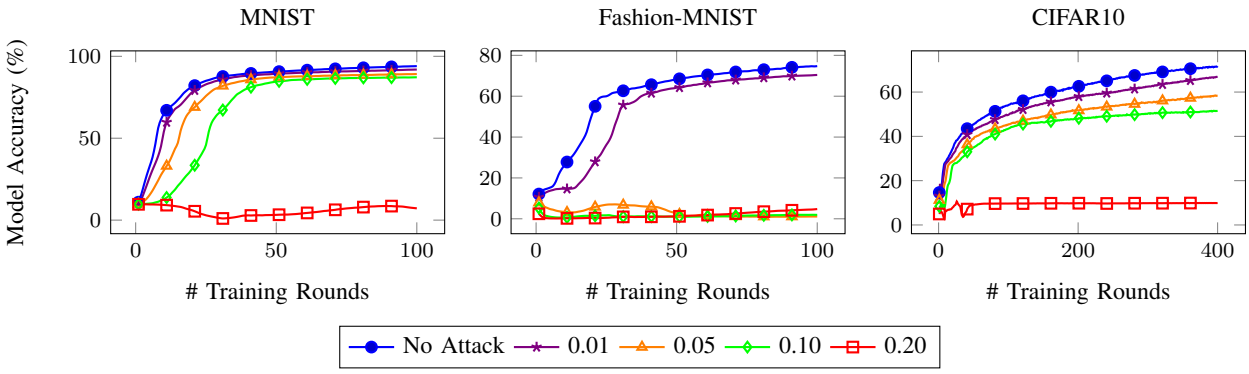

Figure 9: Impact of HIDRA on accuracy with varying ϵ

Backdoor attacks aim to train models that misclassify inputs that have planted trigger patterns [1], [41], [42]. These attacks also bias the aggregate gradient of the model, however, they do not aim to create optimal bias. Therefore, existing strong robust aggregators have been shown to mitigate these attacks [14]. Untargeted poisoning attacks, our motivating application, also create bias but differ in their goal, i.e., to destroy model performance [20]. The unifying vulnerability across poisoning attacks is that they bias gradient vectors used in averaging, with the malicious ones having orientation and magnitude different from the benign vectors [55]. Knowing something about the distribution of benign vectors, therefore, is a universal basis to build defenses. The intuition that outlier removal based on the distributional properties of benign vectors can increase the robustness of modern ML models is evident in early work [29], but many such defenses lack theoretical analysis and evaluation on non-adaptive attacks. The main issue is how to provably defeat an adaptive attacker, one which carefully adjusts the attack strategy with knowledge of the defense.

Weak robust aggregators. Many aggregators are accompanied by analyses under adaptive worst-case inputs. They have weak bounds that are proportional to $O(\sqrt{\epsilon \cdot d})$. Yin et al. proposed coordinate-wise trimmed mean and median for byzantine robust federated learning [9]. Concurrently, defenses based on Euclidean distances between vectors, Krum, and geometric median, were proposed [10], [12] and subsequently extended [56]. In Krum, for each vector, the sum of distances of the closest $n - n\epsilon - 1$ vectors is computed, and the vector with the smallest such aggregate distance is chosen as the mean. Analyses for all of these aggregators give a bias upper bound of $O(\sqrt{\epsilon \cdot d})$ [23], [31].

Poisioning attacks against weak robust aggregators. Untargeted poisoning attacks, especially in federated learning, are the prime nemesis for robust aggregators. Bit flipping and label flipping were early attacks that proved to be effective without any defense in place [9], [57]. Several weak robust aggregators such as coordinate-wise, trimmed mean, and Krum have demonstrably mitigated these attacks [9], [12]. Xie et al. [3] and Fang et al. [2] have proposed several attacks that mitigate known weak robust aggregators as well. They follow a similar strategy of placing corrupted gradients to ours along the opposite direction of the benign gradient and tuning the magnitude of the corrupted gradients to defeat the specific attacks considered. Their bias introduced, however, is far from optimal. Shejwalkar et al. have recently shown that these attacks do not affect the training process much at all when the corruption fraction is low ($\epsilon < 0.01$) [58]. Nevertheless, all of these attacks have been recently shown to be mitigated by much stronger defenses, even at higher $\epsilon \in [0.1 - 0.5)$. This is why our focus has been on strong robust aggregators.

Strong robust aggregators. These aggregators achieve an $\tilde{O}(\sqrt{\epsilon})$ upper bound on the bias, thus, removing the dependence on d [14], [18], [59] (see Section 3 for details). Zhu et al. show that these aggregators when implemented in practice mitigate all aforementioned weak robust aggregators. We design HIDRA to break the guarantees offered strong defenses [14], [18], [59]. We give optimal attacks in the low dimensional regime and identify a common computational bottleneck in them. We are not aware of prior work that carefully explains why the bottleneck is somewhat fundamental and identifies its existence as an exploit opportunity. The state-of-the-art practical realization of strong robust aggregator is given by [14]. Our attack creates bias proportional to $\sqrt{d}$ in these practical realizations. Our results leave the gap wide open between the ideally desired robustness and that which is practical presently in high dimensional settings.

Robust mean estimation. Robust mean estimation in high dimensions lends a theoretical underpinning to byzantine robust federated learning, as explained in Section 2. Apart from the robust aggregators we discussed in Section 3, Zhu et al. propose a robust aggregator using Generative Adversarial Networks that are trained to remove the outliers. This method relies heavily on training and tuning the hyperparameters of GANs to provide to optimal bias guarantees which is not feasible in practical scenarios [14]. Cheng et al. proposed a robust mean aggregator based on Filtering with optimal bias guarantees with time complexity $O(\frac{nd}{\epsilon^6})$

[60]. This aggregator relies on solving a dual SDP problem with 2-degree constraints in linear time. However, we are not aware of any such efficient SDP solvers for high d. Another work proposes a robust aggregator with optimal bias guarantees with $O(nd \cdot poly(\log d))$ [61]. The analysis of this aggregator relies on having auxiliary information about the number of corrupted directions.

7. Conclusion & Future Work

We have shown nearly optimal attacks against practical realizations of strong robust aggregators. In our experiments, untargeted poisoning attacks using our approach against these algorithms almost completely destroy model performance where previous attacks fail to have much impact. Strong aggregators are thus practically much more vulnerable than anticipated by prior theoretical analysis when working with high-dimensional vectors.

We have argued that the vulnerability is fundamental to strong aggregators that are deterministic and aim to work for all feasible distributions generically. Future work can explore provable algorithms to directly improve the computational bottlenecks we highlight, consider randomized defenses, or specialize for features arising in certain gradient distributions. HIDRA is not meant to be a stealthy attack; a specialized defense targeting HIDRA can detect its signature. Devising stealthier attacks that next adaptation of defenses cannot detect would be interesting. Extending HIDRA to other poisoning attacks is another possibility.

Acknowledgements

We are thankful to the anonymous reviewers and our shepherd on the program committee. We also wish to thank Ankit Pensia, Kareem Shehata, and Jason Zhijingcheng Yu for their helpful feedback on previous drafts. This research is supported by the research funds of the Crystal Centre at National University of Singapore and the Ministry of Education Singapore grants: Tier-2 grant MOE-T2EP20220-0014 and Tier-1 grant T1 251RES2023. All opinions expressed in the work are those of the authors.

References

[1] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *ICLR*, 2019.

[2] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *USENIX Security*, 2020.

[3] C. Xie, O. Koyejo, and I. Gupta, "Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation," in *UAI*, 2020.

[4] F. Tramèr, R. Shokri, A. San Joaquin, H. Le, M. Jagielski, S. Hong, and N. Carlini, "Truth serum: Poisoning machine learning models to reveal their secrets," in *CCS*, 2022.

[5] D. Solans, B. Biggio, and C. Castillo, "Poisoning attacks on algorithmic fairness," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020.

[6] S. Zhang, H. Yin, T. Chen, Z. Huang, Q. V. H. Nguyen, and L. Cui, "Pipattack: Poisoning federated recommender systems for manipulating item promotion," in *WSDM*, 2022.

[7] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992.

[8] J. W. Tukey, "A survey of sampling from contaminated distributions," *Contributions to probability and statistics*, 1960.

[9] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *ICML*, 2018.

[10] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, 2022.

[11] X. Chen, T. Chen, H. Sun, S. Z. Wu, and M. Hong, "Distributed training with heterogeneous data: Bridging median-and mean-based algorithms," *NeurIPS*, 2020.

[12] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *NeurIPS*, 2017.

[13] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *NDSS*, 2021.

[14] B. Zhu, L. Wang, Q. Pang, S. Wang, J. Jiao, D. Song, and M. I. Jordan, "Byzantine-robust federated learning with optimal statistical rates," in *AISTATS*, 2023.

[15] I. Diakonikolas and D. M. Kane, "Recent advances in algorithmic high-dimensional robust statistics," 2019.

[16] E. Amaldi and V. Kann, "The complexity and approximability of finding maximum feasible subsystems of linear relations," *Theoretical computer science*, 1995.

[17] I. Diakonikolas, G. Kamath, D. M. Kane, J. Li, A. Moitra, and A. Stewart, "Being robust (in high dimensions) can be practical," in *ICML*, 2017.

[18] S. Hopkins, J. Li, and F. Zhang, "Robust and heavy-tailed mean estimation made simple, via regret minimization," *NeurIPS*, 2020.

[19] P. K. Kothari and D. Steurer, "Outlier-robust moment-estimation via sum-of-squares," 2017.

[20] L. Muñoz González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *AISec*, 2017.

[21] J. J. Cuppen, "A divide and conquer method for the symmetric tridiagonal eigenproblem," *Numerische Mathematik*, 1980.

[22] P. Arbenz, D. Kressner, and D. Zürich, "Lecture notes on solving large scale eigenvalue problems," *D-MATH, EHT Zurich*, 2012.

[23] K. A. Lai, A. B. Rao, and S. Vempala, "Agnostic estimation of mean and covariance," in *IEEE FOCS*, 2016.

[24] P. J. Huber, *Robust statistics*. John Wiley & Sons, 2004, vol. 523.

[25] I. Diakonikolas, G. Kamath, D. Kane, J. Li, A. Moitra, and A. Stewart, "Robust estimators in high-dimensions without the computational intractability," *SIAM Journal on Computing*, 2019.

[26] P. K. Kothari, J. Steinhardt, and D. Steurer, "Robust moment estimation and improved clustering via sum of squares," in *STOC*, 2018.

[27] P. K. Kothari, P. Manohar, and B. H. Zhang, "Polynomial-time sum-of-squares can robustly estimate mean and covariance of gaussians optimally," in *ALT*, 2022.

[28] L. Bottou *et al.*, "Stochastic gradient learning in neural networks," *Proceedings of Neuro-Nımes*, 1991.

[29] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *ACSAC*, 2016.

[30] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *ICML*, 2019.

[31] G. Lugosi and S. Mendelson, "Robust multivariate mean estimation: the optimality of trimmed mean," *Ann. Statist.*, 2021.

[32] S. O. Gharan, "Power method, spectral sparsification," Lecture Notes, 2018, cSE 521: Design and Analysis of Algorithms I, University of Washington.

[33] I. Diakonikolas and D. M. Kane, in *Algorithmic high-dimensional robust statistics*. Cambridge university press, 2023.

[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

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

[36] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:18268744

[37] "HIDRA," 2024. [Online]. Available: https://github.com/sarthak-choudhary/HIDRA

[38] R. A. Maronna, R. D. Martin, V. J. Yohai, and M. Salibián-Barrera, *Robust statistics: theory and methods (with R)*. John Wiley & Sons, 2019.

[39] J. Newsome, B. Karp, and D. Song, "Paragraph: Thwarting signature learning by training maliciously," in *RAID*, 2006.

[40] G. Wang, T. Wang, H. Zheng, and B. Y. Zhao, "Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers," in *USENIX Security*, 2014.

[41] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS*, 2020.

[42] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," *NeurIPS*, 2020.

[43] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen *et al.*, "{FLAME}: Taming backdoors in federated learning," in *USENIX Security*, 2022.

[44] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[45] C. Wu, X. Yang, S. Zhu, and P. Mitra, "Mitigating backdoor attacks in federated learning," *arXiv preprint arXiv:2011.01767*, 2020.

[46] C. Xie, O. Koyejo, and I. Gupta, "Zenops: A distributed learning system integrating communication efficiency and security," *Algorithms*, 2022.

[47] J. Jia, X. Cao, and N. Z. Gong, "Intrinsic certified robustness of bagging against data poisoning attacks," in *AAAI*, 2021.

[48] J. Jia, Y. Liu, X. Cao, and N. Z. Gong, "Certified robustness of nearest neighbors against data poisoning and backdoor attacks," in *AAAI*, 2022.

[49] A. Levine and S. Feizi, "Deep partition aggregation: Provable defense against general poisoning attacks," *arXiv preprint arXiv:2006.14768*, 2020.

[50] E. Rosenfeld, E. Winston, P. Ravikumar, and Z. Kolter, "Certified robustness to label-flipping attacks via randomized smoothing," in *ICML*. PMLR, 2020.

[51] M. Weber, X. Xu, B. Karlaš, C. Zhang, and B. Li, "Rab: Provable robustness against backdoor attacks," in *IEEE SP*, 2023.

[52] T. Krauß and A. Dmitrienko, "Mesas: Poisoning defense for federated learning resilient against adaptive attackers," in *CCS*, 2023.

[53] C. Xie, Y. Long, P.-Y. Chen, Q. Li, S. Koyejo, and B. Li, "Unraveling the connections between privacy and certified robustness in federated learning against poisoning attacks," in *CCS*, 2023.

[54] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *ESORICS*, 2020.

[55] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitraş, and N. Papernot, "On the effectiveness of mitigating data poisoning attacks with gradient shaping," *arXiv preprint arXiv:2002.11497*, 2020.

[56] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *ICML*. PMLR, 2018.

[57] A. Paudice, L. Muñoz-González, and E. C. Lupu, "Label sanitization against label flipping poisoning attacks," in *ECML PKDD*, 2019.

[58] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *IEEE SP*, 2022.

[59] B. Zhu, J. Jiao, and D. Tse, "Deconstructing generative adversarial networks," *IEEE Transactions on Information Theory*, 2020.

[60] Y. Cheng, I. Diakonikolas, and R. Ge, "High-dimensional robust mean estimation in nearly-linear time," in *ACM-SIAM SODA*, 2019.

[61] Y. Dong, S. Hopkins, and J. Li, "Quantum entropy scoring for fast robust mean estimation and improved outlier detection," *NeurIPS*, 2019.

[62] R. Vershynin, "How close is the sample covariance matrix to the actual covariance matrix?" *Journal of Theoretical Probability*, 2012.

Appendix A.
Theoretical Analysis: Complete Proofs

Here we provide the complete proofs for Lemmas in Sections 4.3 and 4.4.

Lemma 1.1. *Let $\hat{s}$ be any direction, $\mu^{\hat{s}}$ be the component of benign mean along $\hat{s}$, and $\sigma_{\hat{s}}$ be the variance along $\hat{s}$. Then, replacing an $\epsilon \cdot n$ vectors in X with $-z(\hat{s})$, where $z = \sqrt{\frac{\xi - \sigma_{\hat{s}}^2}{\epsilon^2 + \epsilon \cdot (1-\epsilon)^2}} - \mu^{\hat{s}} \pm \frac{\delta \cdot \sigma_{\hat{s}}}{\sqrt{n}}$, results in bias of $\Omega(\sqrt{\epsilon})||\Sigma||^{\frac{1}{2}}$ with probability at least $1 - \exp(-\delta^2)$, for all $\delta > 1$.*

Proof. We are given $X : \{x_1, x_2, \ldots, x_n\}$ and $Y : \{y_1, y_2, \ldots, y_n\}$ and a direction $\hat{s}$. Without loss of generality, let's assume that we corrupt the first $n \cdot \epsilon$ update vectors in the following manner:

$$y_i = -z\hat{s},\ i \in [1, \ldots, \epsilon n] \tag{4}$$

We have to find this value of z.

$$bias = \frac{1}{n}||\sum x_i - \sum y_i||_2 \tag{5}$$

$$\geq \frac{1}{n}\left[\sum \langle x_i, \hat{s}\rangle - \sum \langle y_i, \hat{s}\rangle\right] \tag{6}$$

$$\geq \mu^{\hat{s}} - \mu_c^{\hat{s}} \tag{7}$$

$$\mu_c^{\hat{s}} = \frac{1}{n}\left(\sum_{i=1}^{n\epsilon}\langle y_i, \hat{s}\rangle + \sum_{i=n\epsilon+1}^{n}\langle x_i, \hat{s}\rangle\right) \tag{8}$$

$$= \epsilon(-z) + \frac{1}{n}\sum_{i=n\epsilon+1}^{n}\langle x_i, \hat{s}\rangle \tag{9}$$

$$= \epsilon(-z) + (1-\epsilon)\left(\mu^{\hat{s}} \pm \frac{\delta\sigma_{\hat{s}}}{\sqrt{n}}\right) \tag{10}$$

where $\frac{\delta\sigma_{\hat{s}}}{\sqrt{n}}$ is the error with probability $1 - e^{-\delta^2}$ using chernoff bound, we will add this to z at the end

$$\implies \mu^{\hat{s}} - \mu_c^{\hat{s}} = \epsilon(z + \mu^{\hat{s}}) \tag{11}$$

We find that value of z for which the variance along $\hat{s}$ goes from $\sigma_{\hat{s}}^2$ to ξ and does not exceed it.

$$\frac{1}{n}\left(\sum_{i=1}^{n\epsilon}\left(-z-\mu_c^{\hat{s}}\right)^2+\sum_{i=n\epsilon+1}^{n}\left(x_i^{\hat{s}}-\mu_c^{\hat{s}}\right)^2\right)\leq\xi \quad (12)$$

$$\frac{1}{n}\left(\sum_{i=1}^{n\epsilon}\left(-z-\mu_c^{\hat{s}}\right)^2+\sum_{i=1}^{n}\left(x_i^{\hat{s}}-\mu_c^{\hat{s}}\right)^2-\sum_{i=1}^{n\epsilon}\left(x_i^{\hat{s}}-\mu_c^{\hat{s}}\right)^2\right)\leq\xi \quad (13)$$

We solve for the Equation 14 since it implies 13

$$\frac{1}{n}\left(\sum_{i=1}^{n\epsilon}\left(-z-\mu_c^{\hat{s}}\right)^2+\sum_{i=1}^{n}\left(x_i^{\hat{s}}-\mu_c^{\hat{s}}\right)^2\right)=\xi \quad (14)$$

$$\frac{1}{n}\left(n\epsilon(-z-\mu_c^{\hat{s}})^2+\sum_{i=1}^{n}\left(x_i^{\hat{s}}-\mu^{\hat{s}}+\mu^{\hat{s}}-\mu_c^{\hat{s}}\right)^2\right) \quad (15)$$

$$\frac{1}{n}\left(n\epsilon\left(-z-\mu_c^{\hat{s}}\right)^2+\Sigma_{i=1}^{n}\left(x_i^{\hat{s}}-\mu^{\hat{s}}\right)^2+n\left(\mu^{\hat{s}}-\mu_c^{\hat{s}}\right)^2\right) \quad (16)$$

$$\implies \epsilon\left(-z-\mu_c^{\hat{s}}\right)^2+\sigma_{\hat{s}}^2+\epsilon^2\left(z+\mu^{\hat{s}}\right)^2=\xi \quad (17)$$

after substituting from 11

$$\epsilon\left(-z+\epsilon\left(z+\mu^{\hat{s}}\right)-\mu^{\hat{s}}+\delta\right)^2+\sigma_{\hat{s}}^2+\epsilon^2\left(z+\mu^{\hat{s}}\right)^2=\xi \quad (18)$$

$$\left(z+\mu^{\hat{s}}\right)^2\cdot\left(\epsilon^2+\epsilon(1-\epsilon)^2\right)=\xi-\sigma_{\hat{s}}^2 \quad (19)$$

$$z+\mu^{\hat{s}}=\sqrt{\frac{\xi-\sigma_{\hat{s}}^2}{\epsilon^2+\epsilon(1-\epsilon)^2}} \quad (20)$$

Therefore,

$$z=\sqrt{\frac{\xi-\sigma_{\hat{s}}^2}{\epsilon^2+\epsilon(1-\epsilon)^2}}-\mu^{\hat{s}}\pm\frac{\delta\sigma_{\hat{s}}}{\sqrt{n}} \quad (21)$$

Substituting the value of $z+\mu^{\hat{s}}$ in Equation 11, we can bound the bias as follows.

$$bias\geq\sqrt{\epsilon}\cdot\sqrt{\frac{\xi-\sigma_{\hat{s}}^2}{\epsilon+(1-\epsilon)^2}} \quad (22)$$

$$bias\geq\Omega(\sqrt{\epsilon})\cdot\sqrt{\xi},\ \{\because\xi>9||\Sigma||_2,\sigma_{\hat{s}}<||\Sigma||_2^{\frac{1}{2}}\} \quad (23)$$

□

Lemma 1.2. *Let $\hat{s}$ be in the direction of the benign aggregate mean $\mu=\frac{1}{n}\sum_{i=1}^{n}x_i$ and Y be ϵ-corrupted set of vectors. Let the corrupted vectors be along $-\hat{s}$ with magnitude $z=\sqrt{\frac{\xi-\sigma_{max}^2}{\epsilon^2+\epsilon(1-\epsilon)^2}}-\mu^{\hat{s}}$. Then, $||\Sigma_Y||_2\leq\xi$, where $||\Sigma_Y||_2$ is the spectral norm of the covariance matrix of Y.*

Proof. Consider a random direction $\hat{t}$ and say the corruptions are $-z'\hat{t}$ (notice that z' and $\hat{t}$ are different from z and $\hat{s}$). Let's also consider ξ as the final variance $\bar{\sigma}_{\hat{t}}^2$ that will be attained along the direction $\hat{t}$. Then as long as $\bar{\sigma}_{\hat{t}}^2>\sigma_{max}^2$, the equation 19 from Lemma 1.1 can be rewritten as

$$\bar{\sigma}_{\hat{t}}^2=K\cdot(\mu^{\hat{t}}+z')^2+\sigma_{\hat{t}}^2,\ \{K=\sqrt{\epsilon^2+\epsilon(1-\epsilon)^2}\} \quad (24)$$

Now, say $\hat{s}$ is the direction of benign mean μ and we place the corruptions along $-\hat{s}$ rather than $-\hat{t}$. Therefore,

$$\bar{\sigma}_{\hat{s}}^2=K\cdot(\mu^{\hat{s}}+z)^2+\sigma_{\hat{s}}^2,\ \{K=\sqrt{\epsilon^2+\epsilon(1-\epsilon)^2}\} \quad (25)$$

In this scenario, the corruptions will have a projection along $-\hat{t}$ which with be $z'=\langle-z\hat{s},-\hat{t}\rangle$ and μ will have a component $\mu^{\hat{t}}=\langle\mu,\hat{t}\rangle$ along $\hat{t}$. So, if we add corruptions along $\hat{s}$ then the final variance in the direction $\hat{t}$ can be computed by substituting these projections z' and $\mu^{\hat{t}}$ in equation 22. Hence we get,

$$\bar{\sigma}_{\hat{t}}^2=K\cdot(\langle\mu,\hat{t}\rangle+\langle-z\hat{s},-\hat{t}\rangle)^2+\sigma_{\hat{t}}^2$$
$$\bar{\sigma}_{\hat{s}}^2=K\cdot(\mu^{\hat{s}}+z)^2+\sigma_{\hat{s}}^2$$

Notice that, if we choose $\sigma_{\hat{s}}^2=\sigma_{max}^2$ then,

$$\sigma_{\hat{t}}^2\leq\sigma_{\hat{s}}^2,\ \because\sigma_{\hat{s}}^2=\sigma_{max}^2$$
$$(\langle-z\hat{s},\hat{-t}\rangle+\langle\mu,\hat{t}\rangle)^2\leq(\mu^{\hat{s}}+z)^2$$
$$\implies\bar{\sigma}_{\hat{s}}^2\geq\bar{\sigma}_{\hat{t}}^2,\ \forall\hat{t}$$
$$\implies||\Sigma_Y||_2=\sigma_{\hat{s}}^2\leq\xi$$

□

Lemma 1.3. *If strong robust aggregators are used to robustly aggregate over c chunks, as in Algorithm 6 with a single threshold ξ, then* HIDRA *performed over each chunk, as in Algorithm 5, will result in a bias of $\Omega(\sqrt{\epsilon c})\cdot\sqrt{\xi}$.*

Proof. For each chunk i,

$$bias_i\geq k\sqrt{\epsilon}\cdot\sqrt{\xi}$$

Therefore,

$$bias\geq\sqrt{\sum_{i=1}^{c}(bias_i)}$$
$$bias\geq k\cdot\sqrt{\epsilon c}\cdot\sqrt{\xi}$$

□

Lemma 2.1. *Given an ϵ-corrupted set of vectors Y', as constructed using the elements of construction (EC), the direction of maximum variance of Y' lies with a small angle of $cos^{-1}\left(\sqrt{1-O(\frac{\delta^2}{n})}\right)$ from the difference of vectors in set B with probability at least $1-2\exp(\frac{-\delta^2}{2})$, for all $\delta>2$.*

Proof. For set $Y = \{y_1, \ldots, y_{n-n\epsilon}\}$ and $Y' = \{y_1, \ldots, y_{n-n\epsilon}, y_{n-n\epsilon}, \ldots, y_n\}$. Consider,

$$\mu = \frac{1}{n-n\epsilon}\sum_{i=1}^{n-n\epsilon} y_i = \hat{\mu} \tag{26}$$

$$\mu' = \frac{1}{n}\sum_{i=1}^{n} y_i = (1-\epsilon)\hat{\mu} + \frac{1}{n}\sum_{i=n-n\epsilon+1}^{n} y_i \tag{27}$$

$$\mu' = (1-\epsilon)\hat{\mu} + \mu_c \quad \text{where } \mu_c = \frac{1}{n}\sum_{i=n-n\epsilon+1}^{n} y_i \tag{28}$$

Each y_i for $i \in [n-n\epsilon+1, n]$ aligns along one of the directions in $B = \{b_1, b_2\}$, so

$$\mu_c = \frac{1}{n}\left(\sum_{j=1}^{\frac{n\epsilon}{2}}(\sqrt{d}+l_{1j})\cdot b_1 + \sum_{j=1}^{\frac{n\epsilon}{2}}(\sqrt{d}+l_{2j})\cdot b_2\right) \tag{29}$$

$$\mu_c = \frac{\epsilon\sqrt{d}}{2}(b_1+b_2) \tag{30}$$

Given the set B is chosen randomly without any correlation to Y, without loss of generality, we can consider B as unit vectors along any 2 axes in $\mathbb{R}^d$. For simplicity, we consider B as the unit vectors along the first 2 axes. Following this, $\mu_c \in \mathbb{R}^d$ has first 2 components as $\frac{\epsilon\sqrt{d}}{2}$ and rest as 0.

$$\mu_c = \left[\frac{\epsilon\sqrt{d}}{2}, \frac{\epsilon\sqrt{d}}{2}, 0, \ldots, 0\right] \tag{31}$$

To find the direction of the maximum variance of Y', first, we compute the covariance matrix of Y' as follows.

$$\Sigma_{Y'} = \frac{1}{n}\sum_{i=1}^{n}(y_i-\mu')^T\cdot(y_i-\mu') \tag{32}$$

Substitute μ' from eq. 28

$$\Sigma_{Y'} = \frac{1}{n}\sum_{i=1}^{n}(y_i-(1-\epsilon)\hat{\mu}-\mu_c)^T\cdot(y_i-(1-\epsilon)\hat{\mu}-\mu_c) \tag{33}$$

To simplify the eq. 33 , consider the following equations.

$$\text{Cov}(Y) = \frac{1}{n-n\epsilon}\sum_{i=1}^{n-n\epsilon}(y_i-\hat{\mu})^T\cdot(y_i-\hat{\mu}) = \hat{I} \tag{34}$$

$$\frac{1}{n-n\epsilon}\left(\sum_{i=1}^{n-n\epsilon} y_i^T\cdot y_i\right) - \hat{\mu}^T\cdot\hat{\mu} = \hat{I} \tag{35}$$

$$\sum_{i=1}^{n-n\epsilon} y_i^T\cdot y_i = (n-n\epsilon)(\hat{I}+\hat{\mu}^T\cdot\hat{\mu}) \tag{36}$$

Further from eq. 28, we can write

$$\sum_{i=1}^{n} y_i = (n-n\epsilon)\hat{\mu} + n\mu_c \tag{37}$$

Now, expand the RHS of eq. 33 by multiplying each term and replacing the following terms using eq. 36 and 37

$$\sum_{i=1}^{n} y_i^T\cdot y_i \to (n-n\epsilon)(\hat{I}+\hat{\mu}^T\cdot\hat{\mu}) + \sum_{i=n-n\epsilon+1}^{n} y_i^T\cdot y_i$$

$$\sum_{i=1}^{n} y_i \to (n-n\epsilon)\hat{\mu} + n\mu_c$$

The simplified expression of $\Sigma_{Y'}$ would be:

$$\Sigma_{Y'} = \frac{1}{n}\sum_{i=n-n\epsilon+1}^{n} y_i^T\cdot y_i - \mu_c^T\mu_c$$
$$+(1-\epsilon)\hat{I} - (1-\epsilon)(\mu_c^T\cdot\hat{\mu} + \hat{\mu}^T\cdot\mu_c) + (1-\epsilon)(\epsilon)\hat{\mu}^T\cdot\hat{\mu} \tag{38}$$

Notice the last three terms of the above expression contain the errors due to sampling from the spherical Gaussian in $\hat{\mu}$ and $\hat{I}$. Next, we write the matrix expression of all terms and bound the variance incurred by the error terms by a constant which is negligible compared to the total. Hence, we show that the unit vector direction that maximizes the variance of $\Sigma_{Y'}$ including the error terms will be approximately close to the exact maximum variance direction computed without them.

As all the corrupted vectors are along one of the first 2 axes in $\mathbb{R}^d$, $\sum_{i=n-n\epsilon+1}^{n} y_i^T\cdot y_i$ is a diagonal matrix with the first 2 as non-zero and equal to $\sum_{j=1}^{\frac{n\epsilon}{2}}(\sqrt{d}+l_{ij})^2$ for $i \in [1,2]$. Therefore, the first term of eq. 38 can be written as following where $\gamma = \sum_{j=1}^{\frac{n\epsilon}{2}} l_{ij}^2$, for$i \in [1,2]$:

$$\frac{1}{n}\sum_{i=n-n\epsilon+1}^{n} y_i^T\cdot y_i = \left(\frac{\epsilon d}{2}+\frac{\gamma}{n}\right)\begin{bmatrix}1&0&0&\ldots&0\\0&1&0&\ldots&0\\0&0&0&\ldots&0\\\vdots&\vdots&\vdots&\ddots&\\0&0&0&0&0\end{bmatrix} \tag{39}$$

Further, from eq. 31, $\mu_c^T\cdot\mu_c$ has first 2×2 terms as non-zero and equal to $\frac{\epsilon\sqrt{d}}{2}\times\frac{\epsilon\sqrt{d}}{2}$. So, we have:

$$\mu_c^T\cdot\mu_c = \frac{\epsilon^2 d}{4}\begin{bmatrix}1&1&0&\ldots&0\\1&1&0&\ldots&0\\0&0&0&\ldots&0\\\vdots&\vdots&\vdots&\ddots&\\0&0&0&0&0\end{bmatrix} \tag{40}$$

Now, for the sampling error terms in eq. 38, assume the sampling error in the first 2 component of $\hat{\mu}$ is $\hat{\mu}_1, \hat{\mu}_2$. Then, the matrix form of the sampling error terms is as follows:

$$(1-\epsilon)(\mu_c^T\cdot\hat{\mu}+\hat{\mu}^T\cdot\mu_c)$$
$$= \frac{(1-\epsilon)\epsilon\sqrt{d}}{2}\cdot\begin{bmatrix}2\hat{\mu}_1&\hat{\mu}_1+\hat{\mu}_2&0&\ldots&0\\\hat{\mu}_1+\hat{\mu}_2&2\hat{\mu}_2&0&\ldots&0\\0&0&0&\ldots&0\\\vdots&\vdots&\vdots&\ddots&\\0&0&0&0&0\end{bmatrix} \tag{41}$$

As set Y is sampled from a spherical Gaussian, it implies that both $\hat{\mu}_1$ and $\hat{\mu}_2$ is less than $O(\frac{\delta}{\sqrt{n}})$ with probability at least $(1-2e^{\frac{-\delta^2}{2}})$ using Chernoff bound. Using Frobenius norm as bound for Spectral norm, we can claim the variance incurred by this is $\mathrm{O}(\frac{\delta}{\sqrt{n}}) \cdot \sqrt{d}$. Similarly, the variance incurred by the error term $\hat{\mu}^T \cdot \hat{\mu}$ is $O(\frac{\delta^2}{n}) \cdot d$ with probability at least $(1-(\frac{\delta}{\sqrt{n}})^d)$ using Chernoff bound. And for $\hat{I}$, the spectral norm is bounded by $O(\frac{d}{n})$ [62].

Now, we combine the exact and sampling terms in eq. 38 and write them in matrix form where $\alpha = \frac{\epsilon d}{2}(1-\frac{\epsilon}{2}) + \frac{\gamma}{n}$, $\beta = -\frac{\epsilon^2 d}{4}$, and Σ_{err} is the combined covariance matrix for sampling error terms.

$$\Sigma_{Y'} = \begin{bmatrix} \alpha & \beta & 0 & \dots & 0 \\ \beta & \alpha & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \Sigma_{err} \tag{42}$$

Finally, we have a matrix form of the exact covariance term and upper bound of the error terms. Next, take an arbitrary unit vector $v = [v_1, v_2, \dots, v_d]$ and the variance of set Y' along v, denoted by σ_v^2, is as follows:

$$\sigma_v^2 = \langle v, \Sigma_{Y'} \cdot v \rangle \tag{43}$$

$$\sigma_v^2 = \alpha(v_1^2 + v_2^2) + 2\beta(v_1 v_2) + \langle v, \Sigma_{err} \cdot v \rangle \tag{44}$$

Notice the exact part of the variance scales with d which is much larger than the variance incurred by error terms. Since α is positive and β is negative, the exact part is maximum when $v_1^2 + v_2^2$ gets equal to 1 and $v_1 v_2$ attains its minimum value. Hence, the exact part attains its maximum value at $v_1 = \frac{1}{\sqrt{2}}$, $v_2 = -\frac{1}{\sqrt{2}}$ or vice-versa which is the direction of difference of vectors in set B. Now, we argue that the maxima of the whole expression should be close to the above solution. Consider another unit vector $v' \in \mathbb{R}^d$ close to the above solution $v = [\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}]$ such that their dot product is ρ, we can write

$$v_1' - v_2' = \sqrt{2}\rho \tag{45}$$

$$v_1'^2 + v_2'^2 - 2v_1' v_2' = 2\rho^2 \tag{46}$$

Given $\alpha > |\beta|$, the decrease in variance incurred by exact terms is at least $2(1-\rho^2)|\beta|$. v' is the maxima for the whole expression as long this decrease is less than the variance incurred by error terms. As shown earlier, the maximum value of $\langle v, \Sigma_{err} \cdot v \rangle$ is $O(\frac{\delta^2}{n} \cdot d)$ with probability at least $1-2e^{\frac{-\delta^2}{2}}$. For v' to be the direction of maximum variance,

$$2(1-\rho^2)|\beta| \leq O\left(\frac{\delta^2}{n}\right) \cdot d \tag{47}$$

$$2(1-\rho^2)\frac{\epsilon^2 d}{4} \leq O\left(\frac{\delta^2}{n}\right) \cdot d \tag{48}$$

$$\sqrt{1 - \frac{2}{\epsilon^2} O\left(\frac{\delta^2}{n}\right)} \leq \rho \tag{49}$$

Hence, ignoring $\frac{2}{\epsilon^2}$ we have shown that the direction of the maximum variance of Y' has an angle of less than $cos^{-1}\left(\sqrt{1-O(\frac{\delta^2}{n})}\right)$ from the difference of vectors in set B with probability at least $1-2e^{\frac{-\delta^2}{2}}$. □

Lemma 2.2. *Let $f(Y') = \tilde{\mu}$ and $\mu' = \frac{1}{n}\sum_{i=0}^{n} Y'[i]$, then Alg. 7 returns the difference of vectors in set B with probability at least $1 - n \cdot \exp(\frac{-n\delta^2}{2})$, for all $\delta > 1$.*

Proof. Similar to the proof of Lemma 2.1, since the set B is chosen randomly without any correlation to Y, we can take B as any 2 axes in $\mathbb{R}^d$. For simplicity of calculations, let's take B as the first 2 axes. Then, we can write μ' can be written as the following using eq.28 and 31,

$$\mu' = [\frac{\epsilon\sqrt{d}}{2}, \frac{\epsilon\sqrt{d}}{2}, 0, \dots, 0] + (1-\epsilon)\hat{\mu} \tag{50}$$

Notice for all $y_i \in Y'$, $\langle \frac{y_i}{||y_i||_2}, \hat{\mu} \rangle$ follows a 1-D Gaussian distribution with mean 0 and variance $\frac{1}{\sqrt{n}}$. Therefore, it would be $O(\delta)$ with probability of $1 - e^{\frac{-n\delta^2}{2}}$ So, for all corrupted vectors $y_i \in Y'$ with probability $1 - e^{\frac{-n\delta^2}{2}}$,

$$\langle \frac{y_i}{||y_i||_2}, \mu' \rangle = \langle b_j, \mu' \rangle \quad b_j \in B \tag{51}$$

$$\langle \frac{y_i}{||y_i||_2}, \mu' \rangle = \frac{\epsilon\sqrt{d}}{2} \pm (1-\epsilon)O(\delta) \tag{52}$$

For uncorrupted $y_i \in Y'$, consider $y_1 = [y_{11}, y_{12}, \dots, y_{1d}]$,

$$\langle \frac{y_1}{||y_1||_2}, \mu' \rangle = \frac{\epsilon\sqrt{d}}{2||y_1||_2}(y_{11} + y_{12}) + (1-\epsilon)O(\delta) \tag{53}$$

For such uncorrupted y_i in Y', $\frac{\sqrt{d}}{||y_i||_2}$ is a constant $t \approx 1$ since each of them is a random sample from d-dimensional spherical gaussian with mean as 0 and covariance as I, hence each of them has an L_2 norm of $\sqrt{d}$ on expectation.

$$\langle \frac{y_1}{||y_1||_2}, \mu' \rangle = \frac{\epsilon t}{2}(y_{11} + y_{12}) + (1-\epsilon)O(\delta) \tag{54}$$

For the projection in eq. 54 to be more than $\frac{\epsilon\sqrt{d}}{2}$, at least one of the two components should be greater than $\frac{\sqrt{d}}{2t} - O(\delta)$. But all components of an uncorrupted vector in Y', as y_1, follow a single dimensional Gaussian distribution with mean as 0 and variance as 1. Then, using Chernoff bounds for 1-D Gaussian, we can calculate the probability of any of the first 2 components being more than $\frac{\sqrt{d}}{2t} - O(\delta)$ as follows,

$$\Pr(y_{11} \text{ or } y_{12} \geq \frac{\sqrt{d}}{2t} - O(\delta)) \leq 2 \cdot e^{(-\frac{d}{8t^2} + O(\delta^2))} \tag{55}$$

Note that the probability of the event mentioned above is exponentially small in d. Hence, we can claim that the projection of μ' along corrupted vectors in Y is larger compared to the uncorrupted vectors with extremely high probability. Furthermore, given $||f(Y') - \mu||_2 \leq \tau\sqrt{\epsilon}$ for some constant τ, for $\mu = 0$ we can write,

$$\langle \tilde{\mu}, v \rangle \leq \tau\sqrt{\epsilon} \qquad \forall v \in \mathbb{R}^d, ||v||_2 = 1 \tag{56}$$

From eq. 52 and eq. 56, for all corrupted vectors in Y'

$$\langle \mu' - \tilde{\mu}, \frac{y_i}{||y_i||_2} \rangle \geq \left(\frac{\epsilon}{2}\sqrt{d} - \tau\sqrt{\epsilon} + (1-\epsilon)O(\delta) \right) \quad (57)$$

And for all other uncorrupted vectors in Y' with probability $1 - ne^{\frac{-n\delta^2}{2}}$ (ignoring negligibly small terms than $ne^{\frac{-n\delta^2}{2}}$),

$$\langle \mu' - \tilde{\mu}, \frac{y_i}{||y_i||_2} \rangle < \left(\frac{\epsilon}{2}\sqrt{d} - \tau\sqrt{\epsilon} + (1-\epsilon)O(\delta) \right) \quad (58)$$

It implies that if we arrange all the vectors of Y' in the order of projection of $\mu' - \tilde{\mu}$ along them, then top $n \cdot \epsilon$ vectors will be the set of corrupted vectors C with very high probability. Given the corrupted set contains $\frac{n\epsilon}{2}$ along each vector in B. It implies that for each corrupted vector there will be $\frac{n\epsilon}{2}$ other corrupted vectors perpendicular to it. For any such mutually perpendicular corrupted vectors, one of them is along b_1 and the other is along b_2. Then,

$$\frac{c_i}{||c_i||_2} - \frac{c_j}{||c_j||_2} = \pm(b_1 - b_2) \qquad c_i, c_j \in C, \langle c_i, c_j \rangle = 0 \quad (59)$$

This confirms that $\frac{c_i}{||c_i||_2} - \frac{c_j}{||c_j||_2}$ equals to the difference of vectors in set B and hence aligns with the direction of maximum variance of Y' as well (Lemma 2.1). The reduction algorithm 7 creates the set C and finds two perpendicular vectors in C. Finally, the algorithm returns the difference of these vectors normalized by their norm which is equal to the difference of vectors in B. □

Appendix B. Addressing Meta-Review Comments

We address the noteworthy concern 2) raised in the meta-review by explaining why we did not evaluate DnC defense (Alg. 2 in [13]). This is because DnC does not satisfy the definitions of a strong robust aggregator, to the best of our knowledge.

We point readers to our meta Algorithm 1 outlined a framework for strong robust aggregators, featuring a loop (Line 3) for the iterative removal of corrupted vectors. DnC does not have such an iterative step. Note that prior analyses of strong robust aggregators underscore that iterations at least equal to the number of corrupted vectors are necessary to provide provable guarantees in the worst-case scenario (i.e. where all corruptions are mutually orthogonal).

To illustrate, we give a straightforward attack against DnC. The absence of the iterative steps means DnC can only eliminate corrupted vectors aligned with the direction of the maximum eigenvector. It fails to detect corrupted vectors aligned with other eigenvectors, such as the eigenvector with the second-highest variance. This limitation arises because DnC never recalculates eigenvectors after removing corrupted vectors along the eigenvector direction with the largest variance, unlike other existing strong aggregators.

Building on this insight, our attack works as follows: Begin by choosing vector, b_1, of dimension d (the dimension of benign gradients) with binary values $\{0, 1\}$ chosen randomly, and its complement vector, b_2. This strategy ensures that the dot product of b_1 and b_2 is zero, as well as the dot product of any subset of dimensions of b_1 and b_2. Next, place a single corrupted vector in the direction of b_1 from the mean, at a distance of $\beta \cdot ||avg||$, where $||avg||$ represent the average distance of benign vectors from the mean. Subsequently, position the remaining corrupted vectors in the direction of b_2 from the mean, at a distance of $c \cdot \beta \cdot ||avg||$. Select a small value for c to ensure that the maximum eigenvector after corruption aligns closely with the direction of b_1 from the mean. Under this attack, DnC only detects and filters out a single corrupted vector along b_1 from the mean, while marking the remaining vectors as inliers and computing the arithmetic mean with them included. The magnitude of the resulting bias incurred by such a corruption strategy depends on the parameter β chosen by the attacker and can be scaled arbitrarily. We have experimentally verified this claim by considering a subset of 1000 dimensions of gradients while training a CNN on CIFAR10 with $\epsilon = 0.2$, as described in section 5.1. We corrupted this set using the aforementioned attack strategy with $c = 0.02$ and employed DnC to compute its robust mean at one step. Fig. 10 illustrates the bias incurred by the attack on the same set with varying β. It scales up with β and can be further scaled arbitrarily. Furthermore, we confirmed that all vectors along b_2 from the mean are marked as inliers by DnC for all the varying values of β. The code for our attack and our implementation[8] of DnC is provided in [37].

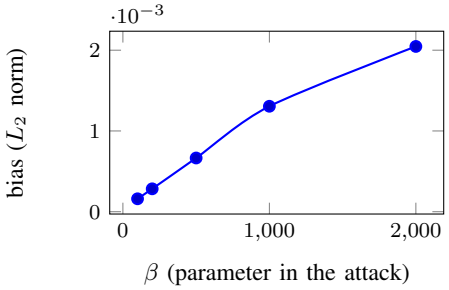

Figure 10: Bias (in L_2 norm) incurred by the proposed attack against DnC with varying β (parameter in the attack).

The DnC defense also proposes to sample a subset of dimensions and considers only them to compute the largest eigenvector to filter outlier vectors in one shot. This is fundamentally different from iterative removal of corrupted vectors by computing a maximum eigenvector in every iteration as done in strong robust aggregators. Our presented attack here will work even if a smaller or larger subset of dimensions are used to compute eigenvectors.

8. The artefacts of the original paper did not have the algorithm's implementation. Hence, we re-implemented it and privately corresponded with one of the authors of that work to check its correctness.

Appendix C. Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

C.1. Summary

This paper introduces HIDRA, a novel attack that targets strong robust aggregators in high-dimensional federated learning environments. Such aggregators provide an upper bound on the bias against poisoning attacks that aim to corrupt a fraction of inputs. However, they introduce a computational bottleneck that limits their application to high-dimensional data like those present in most deep learning settings. To address the bottleneck, practical adaptations split dimensions into disjoint chunks and operate on each chunk individually. HIDRA biases the aggregate of each chunk in the dimensionally split robust aggregators, and the paper shows analytically and experimentally that the resulting total bias approaches theoretical upper bounds and degrades model performance.

C.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field
- Identifies an Impactful Vulnerability

C.3. Reasons for Acceptance

1) HIDRA reveals a new vulnerability in the practical implementation of strong robust aggregators.
2) The paper provides solid theoretical analysis and empirical results that match the theory.
3) The results demonstrate that HIDRA achieves near-optimal bias against robust aggregators in untargeted poisoning attacks.

C.4. Noteworthy Concerns

1) The paper does not propose countermeasures or mitigation strategies against HIDRA.
2) The performance of HIDRA against DnC-based federated learning, a potential defense, is not evaluated.
3) The evaluation focuses on a single architecture, and the impact that this has on the severity of the attack remains uncertain.

Appendix D. Response to the Meta-Review

Thank you reviewers for providing helpful reviews. We have addressed the meta-review comment 2) above regarding DnC in our Appendix B, showing that it does not satisfy the guarantees of a strong aggregator. We leave addressing the remaining comments for future work.