01 Jan 2022

Securing Federated Learning Against overwhelming Collusive Attackers

Priyesh Ranjan

Ashish Gupta

Federico Corò

Sajal K. Das
Missouri University of Science and Technology, sdas@mst.edu

Recommended Citation

P. Ranjan et al., "Securing Federated Learning Against overwhelming Collusive Attackers," *2022 IEEE Global Communications Conference, GLOBECOM 2022 - Proceedings*, pp. 1448 - 1453, Institute of Electrical and Electronics Engineers, Jan 2022.
The definitive version is available at https://doi.org/10.1109/GLOBECOM48099.2022.10000830

Securing Federated Learning against Overwhelming Collusive Attackers

Priyesh Ranjan, Ashish Gupta, Federico Corò, and Sajal K. Das
Department of Computer Science, Missouri University of Science and Technology, Rolla, USA
{pr8pf, ashish.gupta, federico.coro, sdas}@mst.edu

***Abstract*—In the era of a data-driven society with the ubiquity of Internet of Things (IoT) devices storing large amounts of data localized at different places, distributed learning has gained a lot of traction, however, assuming independent and identically distributed data (iid) across the devices. While relaxing this assumption that anyway does not hold in reality due to the heterogeneous nature of devices, federated learning (FL) has emerged as a privacy-preserving solution to train a collaborative model over non-iid data distributed across a massive number of devices. However, the appearance of malicious devices (attackers), who intend to corrupt the FL model, is inevitable due to unrestricted participation. In this work, we aim to identify such attackers and mitigate their impact on the model, essentially under a setting of bidirectional label flipping attacks with collusion. We propose two graph theoretic algorithms, based on Minimum Spanning Tree and k-Densest graph, by leveraging correlations between local models. Our FL model can nullify the influence of attackers even when they are up to 70% of all the clients whereas prior works could not afford more than 50% of clients as attackers. The effectiveness of our algorithms is ascertained through experiments on two benchmark datasets, namely MNIST and Fashion-MNIST, with overwhelming attackers. We establish the superiority of our algorithms over the existing ones using accuracy, attack success rate, and early detection round.**

***Index Terms*—Attackers, federated learning, label flipping**

I. Introduction

The proliferation of smartphones and IoT devices with significant computing capabilities has led to a steep growth in the adoption of machine learning techniques in our daily routine. These devices generate a large amount of data, traditionally processed at a remote server, causing a waste of bandwidth and exposing the privacy of the users as the data may include sensitive information. To address these issues, Google researchers came up with a distributed learning paradigm, called Federated Learning (FL) [1], in which multiple devices (or clients) can collaborate to produce an accurate and generalized model, usually in the presence of a remote server while keeping the data private. The concept involves training a model using the individual data fragments of local devices, followed by passing that model (essentially weights/parameters) to the server for aggregation. Thereafter, the updated model, also referred to as the global model, is relayed back to the clients which marks the end of a single round of the FL process.

However, the lack of transparency invites adversaries who may pose as participants with the intention of corrupting the process by supplying poisoned local models to the server. It not only reduces the performance of the global model but also influences its convergence. Such an adversarial attack can have multi-fold objectives that include promoting the outcome of a particular class by flipping the labels of the corresponding class [2], [3], poisoning the data by adding backdoor elements [4] in their local data shards, or providing random parameters (i.e., model replacement) thus diverging the model from optimal solution [5], [6]. Figure 1 illustrates an example scenario for the digit classification task using images of handwritten digits where two of the participating clients try to inject corruption by flipping the labels of images from 0 to 1 and vice-versa. In this case, the server needs to employ an appropriate attacker detection algorithm to identify these colluding attackers during the aggregation process. Once detected, their models may be excluded from the aggregation to neutralize the impact of the attackers.

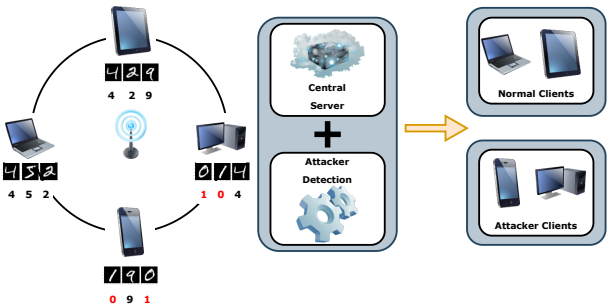


Fig. 1: Illustrating FL setup with label flipping attacks on the local data shards by changing label 0 to 1 and vice-versa. The attacker detection enables server to distinguish normal client from attackers.

In recent years, FL has spurred an active research stream to defend against possible attacks, which can be broadly categorized into two types: (i) untargeted – adversary aims to influence the convergence of global model by corrupting the whole local model [5], [7], [8]; some attacks in this category include random noise addition to local model/gradients, sign flipping, or model replacement, and (ii) targeted – adversary attempts to misclassify specific set of samples (mostly belong to one particular class) while minimally affecting the model performance on other classes [2]–[4], [6], [9]; in this category, the common attacks are label flipping and backdoor. Further, some researchers have focused on robust aggregation methods such as Krum and Multi-Krum [5], median and trimmed median [10], and GoeMed [11].

• **Motivation:** Our work is motivated by the following limitations of existing works. (i) The robust aggregation methods [5], [10]–[12] mostly extended the stochastic gradient descent

978-1-6654-3540-6/22 © 2022 IEEE

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10000830

(SGD) to aggregate the local models while assuming independent and identically distributed (IID) data across the clients, however, in the FL, the heterogeneous nature of devices produces non-IID data. Moreover, these methods can only minimize the adverse effect on the global model but do not mitigate the effect fully. On the flip side, our work primarily focuses on *complete mitigation of the attackers' impact* by excluding their models from aggregation. (ii) The existing methods that detect targeted attacks can work accurately only when the number of attackers is less than the number of normal clients. Though FoolsGold [2] has overcome this limitation, it requires many FL rounds, thus delaying the detection process, and meanwhile, the attackers keep injecting the corruption.

In this work, we address the problem: *how to secure FL against collusive attackers posing label flipping attacks?* To solve this, we propose two graph theoretic algorithms exploiting Maximum Spanning Tree (MST) and k-Densest graph problems. Particularly, we make the following contributions:

- We propose two novel attacker detection algorithms, called MST-AD and Density-AD, by leveraging the correlation computed over the gradients [1] of the clients. Since collusive attackers have a common objective, their models are highly correlated and have the potential to reveal their presence through MST and k-densest graph.
- By incorporating MST-AD and Density-AD in the aggregation, we enable the server to identify the poisoned local models and exclude them.
- We experimentally evaluate the effectiveness of the proposed algorithms on two benchmark image classification datasets with evidence of their superiority over three different existing algorithms.

The rest of the paper is organized as follows. Section II describes our FL setup along with the considered threat model. Section III proposes the attacker detection algorithms while Section IV evaluates these algorithms on two benchmark datasets. Finally, the paper is concluded in Section V.

II. Problem Description

We consider a standard FL setup with a central server and $\mathcal{C}$ clients of which $\mathcal{M}$ clients are attackers (i.e., malicious in nature). Each client c_i possesses a local training data shard $D_i = \{\mathbf{X}, \mathbf{y}\}$ where $\mathbf{X}$ denotes the set of training samples with labels $\mathbf{y}$. For a classification task, the server initializes a global model W^t for round $t = 1$ and dispatches it to all the clients who retrain this model on their local data. Let δ_i^t be the gradients (weights update, i.e., $W^t - w_i^t$) obtained by client c_i, which is sent back to the server in round t. To this end, the server does aggregation as

$$W^{t+1} = W^t + \sum_{i=1}^{\mathcal{C}-\mathcal{M}} p_i \delta_i^t + \sum_{j=1}^{\mathcal{M}} p_j \delta_j^t, \tag{1}$$

where p_i is the weight of client c_i computed over the percentage of data samples the client possesses, and $\sum_i p_i = 1$.

Objective: To mitigate the effect of attackers on global model, the term $\sum_{j=1}^{\mathcal{M}} p_j \delta_j^t$ should be nullified. To achieve this, we aim to correctly identify all $\mathcal{M}$ attackers by leveraging the correlation between δ_i and δ_j for each pair of clients and $i \neq j$.

Assumptions: Our FL setup assumes on following: (i) data across participants follow non-IID distribution; (ii) no client has access to the local model of others; (iii) client has no control over the aggregation algorithm; (iv) the number of attackers is at least 2 to realize collusion case.

Threat model: Our threat model is limited to targeted attacks done by compromised devices. Particularly, we focus on the label flipping attacks posed by the colluding devices even when they overwhelm normal clients. Attackers manually change a particular label (say 'A') to another label (say 'B') and vice-versa, in their local datasets before training the local model. Prior works [2], [13] have also demonstrated that the colluding devices can promote the poisoning effect rapidly.

Attacker Capabilities: The adversary has full control of the compromised devices, however, can not access the local data or model of the benign devices.

III. Proposed Algorithms

In this work, we propose two attacker detection algorithms, MST-AD and Density-AD, essentially named after the underlying graph theory concepts, by leveraging the correlation between the clients' gradients. Since the attackers train on a poisoned dataset (with flipped labels), they should stay closer to each other, i.e., their gradients should show higher similarity in some spaces. Through extensive experiments, we found that the correlation between the clients' gradients can effectively separate collusive attackers from normal clients. We define the correlation between any two clients i and j as

$$r_{ij} = \frac{\sum_d (\delta_i - \bar{\delta})(\delta_j - \bar{\delta})}{\sqrt{\sum_d (\delta_i - \bar{\delta})^2 \ \sum_d (\delta_j - \bar{\delta})^2}}, \tag{2}$$

where $\bar{\delta}$ represents the mean gradients and d is the dimension of the gradients' matrix (the length of the weight vector).

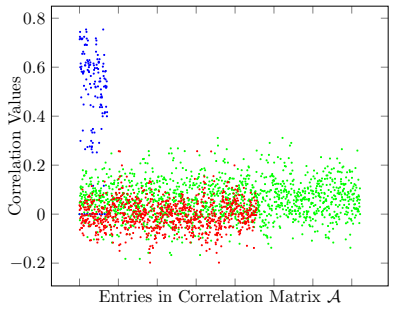


Fig. 2: Correlation values between Attacker-Attacker (blue), Normal-Normal (green) and Attacker-Normal (red).

An empirical observation: By considering an FL setup with 50 clients out of which 25% are attackers, we experiment on Fashion-MNIST [14] dataset with non-IID data, and the correlations between the clients are shown in Fig. 2. It is easy to see that the correlation between the attackers is always greater than that between two normal clients, which in turn is greater than the correlation between an attacker and a normal client. Though the above statement does not hold for every

[1]The terms "gradients" and "weight updates" are used interchangeably.

single correlation, it suffices to distinguish attackers from normal clients using correlation values.

To this end, to design our algorithms we make the following *assumption about the correlation* – given a set of clients $\mathcal{C}$ and a set of attackers $\mathcal{M} \subset \mathcal{C}$, we have the inequality

$$r_{ip} < r_{ij} < r_{pq}, \tag{3}$$

where $p, q \in \mathcal{M}, p \neq q$ and $i, j \in \mathcal{C} \setminus \mathcal{M}, i \neq j$.

By using Eq. (2), we define a correlation matrix $\mathcal{A} \in \mathbb{R}^{n \times n}$ where an entry $\mathcal{A}_{ij}$ corresponds to the correlation coefficient r_{ij} between the clients i and j with $r_{ii} = 0$. This lets us create a graph with n vertices corresponding to the n clients participating in FL and the edge weight between a pair of clients i and j corresponds to the entry $\mathcal{A}_{ij}$. The symmetric nature of the matrix makes the graph a complete undirected graph.

Fig. 3 shows a representative graph for an FL Setup with 6 clients labelled $\{c_1, c_2, c_3, c_4, c_5, c_6\} \in \mathcal{C}$. The clients c_2, c_5, and c_6 are collusive attackers poisoning the model and thus the set of attackers $\mathcal{M} = \{c_2, c_5, c_6\}$. The remaining clients belong to the set of normal clients and given as $\mathcal{C} \setminus \mathcal{M} = \{c_1, c_3, c_4\}$. The gradients with clients are provided for representation purposes only and the correlation values between these gradients are shown as edge weights. As the clients c_2, c_5, and c_6 are attackers, the edge between these clients has a higher value as compared to the other edges in the graph. Similarly, the edges between the attacker and normal client (e.g., between c_2 and c_1) have a lower value as compared to the edges between two normal clients.

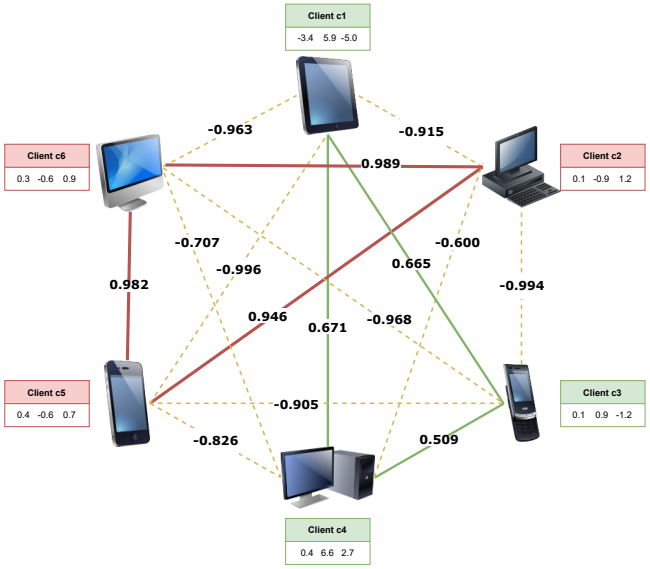


Fig. 3: Example graph showing the correlations (weight on edges) between clients. Attacker-Attacker edges (red) with a higher weight, Attacker-Normal edges (dashed yellow) with a lower weight, and Normal-Normal edge (green). Clients c_2, c_5, and c_6 are the attackers. Numeric values with each client are representing the gradients.

A. MST-AD Algorithm

In this section, we exploit the graph realization obtained from the correlation matrix to create an MST, similar to our previous work [15], which we leverage to distinguish attackers from normal clients. We recall that an MST is a spanning tree of a weighted graph having maximum weight, i.e., on a set of n clients, the tree is composed of $n-1$ edges of maximum weight, subject to a standard constraint that the selected edges do not form a cycle.

Specifically, from the set of edges in the graph, an edge $edge$ is chosen and added to the tree $trees$ if $edge$ has the maximum weight among all the remaining edges in the graph and $edge$ does not form a cycle on the edges of $trees$. Following this, the $edge$ is discarded from the graph, and the edge with the next highest weight is chosen and the process continues till the MST is created. Upon the creation of the corresponding MST, the edge with the lowest weight is chosen and discarded, which results in two sub-trees; the one with the higher average edge weight corresponds to attackers whose gradients are later excluded from the aggregation to mitigate their impact.

Algorithm 1 illustrates the above procedure for creating the sub-trees (essentially MST) over the set of clients. The algorithm starts by initializing the list of trees as an empty set and sorting the edges of the graph in non-decreasing order of their weights (Line 3). We pick the first $n-1$ edges from the sorted set and add them to the tree (Lines 4–7). Considering that our assumption (inequality defined in Eq. 3) holds, the edges connecting the attackers should be included in the MST. Moreover, there would exist a single Attacker-Normal edge having the lowest weight among all the edges in formed MST. The deletion of such edge results in two sub-trees. Since the edges with higher weights exist between the attackers, they would form a single connected sub-tree.

Algorithm 1: MST-AD Algorithm

1 **Input:** Correlation matrix $\mathcal{A}$
2 **Output:** Set of attackers (Atk)
3 $trees \leftarrow \emptyset$; $i \leftarrow 0$; $\mathcal{E} \leftarrow$ sorted edges in non-increasing order
4 **while** $|trees| < n-1$ **do**
5 **if** $\mathcal{E}[i]$ *does not form cycle in* $trees$ **then**
6 $trees \leftarrow trees \cup \mathcal{E}[i]$
7 i++
8 $subT_1, subT_2 \leftarrow$ Remove lowest weighted edge from $trees$
/* Let avg_weight(·) computes average weight of tree */
9 **if** $avg\_weight(subT_1) > avg\_weight(subT_2)$ **then**
10 $Atk \leftarrow subT_1$
11 **else**
12 $Atk \leftarrow subT_2$
13 **return** Atk

***Theorem** 1: Given a correlation matrix $\mathcal{A}$, let the inequality (Eq. 3) hold for any pair of clients, then Algorithm 1 returns the complete and correct set of collusive attackers.*

Proof 1: Given the inequality $r_{ip} < r_{ij} < r_{pq}$ for any pair of attackers $p, q \in \mathcal{M}$, and any pair of normal clients $i, j \in \mathcal{C} \setminus \mathcal{M}$. Let $\mathcal{E}$ be the set of ordered edges of the graph induced by $\mathcal{A}$. Indeed, $\mathcal{E}$ is divided into three contiguous subgroups, the group of edges between any pair of attackers, followed by the group of edges between any pair of normal clients, followed by the last group formed by the edges between normal and

attacker. Then the proof follows directly from the construction of the MST. In fact, by definition of MST, we have to select $n-1$ edges from $\mathcal{E}$ starting from the edges with maximum weights. Following the MST construction, we will have a sub-tree composed of all the malicious clients (edges with higher weight), one sub-tree composed of all the normal clients (the second group of edges in $\mathcal{E}$), and, finally, one single edge (the one with lower weight in such a tree) between the two sub-trees. Thus, by removing the edge with the lowest weight, we are able to distinguish between normal and attacker clients.

B. Density-AD Algorithm

This section introduces another detection algorithm by leveraging the concept of k-densest graph which essentially is a maximum density sub-graph with exactly k vertices. Assuming the inequality defined in Eq. 3 holds, our problem can be realized as a k-densest sub-graph problem in which the objective is to find the k vertices with the highest average weighted degree. Note that the value of k in our problem (i.e., the number of attackers) is not known in advance. Instead, we aim to find out the k vertices whose removal maximizes the density of the remaining sub-graph. The density of a graph is defined as the average of all weighted degrees of the vertices. Given the correlation matrix $\mathcal{A}$, the graph density can be formally defined as

$$density(\mathcal{A}) = \frac{2\sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}}{n(n-1)}. \tag{4}$$

***Definition** 1 (**Sparse vertex**): A vertex v of a graph $\mathcal{G}$ is sparse if its removal increases the density of the graph $\mathcal{G}-v$.*

Our algorithm iterates over each vertex of the graph to identify whether it is a *sparse* vertex or not. If a vertex is sparse, it is removed permanently from the graph otherwise it is replaced back in that graph. Once all the vertices are traversed through successive iterations, the remaining sub-graph with k vertices corresponds to the potential attackers because they have the highest correlations among themselves.

Algorithm 2: Density-AD Algorithm

```
1  Input: Correlation matrix A
2  Output: Set of attackers (Atk)
3  sparse_list ← ∅
4  while i = n down to 1 do
5      B ← A \ A[i]
6      if density(B) > density(A) then
7          sparse_list ← sparse_list ∪ A[i]
8          A ← A \ A[i]
9  if density(sparse_list) > density(A) then
10     Atk ← sparse_list
11 else
12     Atk ← A
13 return Atk
```

The overall steps of the proposed Density-AD algorithm are reported in Algorithm 2. First the list $sparse\_list$ is created for storing the attackers detected during each iteration. The loop at Line 4 iterates n times to testify the sparse nature of each vertex. The set $\mathcal{B}$ is temporarily used to store the elements of set $\mathcal{A}$ (Line 5) excluding the i^{th} vertex. If the i^{th} vertex is a sparse vertex in the graph, the corresponding density of $\mathcal{B}$ will be higher than the density of $\mathcal{A}$ (Line 6) and i^{th} vertex is then appended to the list $sparse$ and subsequently removed from the set $\mathcal{A}$ (Line 8). Next, the density of the nodes in the $sparse$ list is compared with the density of the remaining nodes in $\mathcal{A}$ (Line 10- 12), and the set with a higher density is marked as the set containing the colluding attackers.

IV. Experimental Evaluation

In this section, we evaluate the effectiveness of our proposed attacker detection algorithms and analyze the obtained results with a critical comparison with popular existing algorithms.

A. Experimental Setup

We consider the task of image classification using deep neural networks consisting of 2 Convolutional Neural Network [16] layers followed by 3 fully-connected layers. We use two benchmark datasets: MNIST [17] and Fashion-MNIST (FMNIST in short) [14], each comprising of 60000 training and 10000 testing greyscale images divided equally in 10 classes. For each dataset, the samples were randomly partitioned into $n = 50$ disjoint subsets and each of that is assigned to a single client. Inspired by [6], we adopt Dirichlet distribution with parameter $\alpha = 0.9$, for the partitioning. We simulate the attacker as follows: for the MNIST dataset, the labels of all images with '0' and '1' are flipped and for the FMNIST dataset, labels of all images of "T-Shirt" and "Trouser" are flipped. Note that the adopted label flipping is bi-directional. Further, inspired by [2], we abbreviate the attack scenarios as $A{-}m$ attacks where m is the percentage of attackers to the total number of clients and A is the shorthand for the term 'attack'. For instance, an A–5 attack would refer to the scenario with 5% of the total clients as collusive attackers.

B. Performance Metrics

The evaluation is carried out on the test data while comparing our algorithms with competitive detection algorithms. We consider FoolsGold [2] and GeoMed [10] algorithms for comparison. Besides, our experiments also included federated averaging (FedAvg) [1] as the baseline. We employ the following metrics to quantify the performance: (i) **Test accuracy**, the proportion of correctly classified samples in the test set; (ii) **Attack Success Rate (ASR)**, the proportion of the targeted samples incorrectly classified in the test set. In the context of label flipping, the value corresponds to the ratio of the number of misclassified flipped labels to the total number of labels flipped by adversaries [2]. (iii) **ED**, the earliest round at which all the attackers got detected correctly.

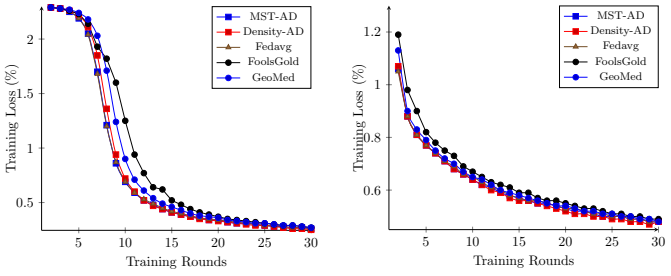
C. Results

While reporting the experimental results in this section, we mainly attempt to answer the following questions: (i) How does the training loss decrease over 30 FL rounds? (ii) What

is the impact of colluding attackers on the test accuracy and F1 Score of all the algorithms? (iii) How efficiently and early do the proposed algorithms detect the label-flipping attackers?

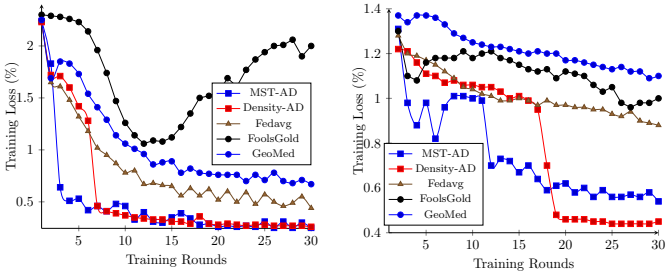
1) Training loss over FL rounds: Figs. 4 and 5 report the obtained loss on the training data over the communication rounds on both the datasets for $A-5$ and $A-70$ attacks, respectively. The effects of collusion can be seen with the algorithms employing central measures (FedAvg and GeoMed) showing higher loss as compared to MST-AD and Density-AD. While the losses for all the algorithms converge in $A-5$ attack, it diverges a lot for the existing algorithms when the attackers overwhelm the normal clients, i.e., in the case of $A-70$ attack. As the number of attackers increases, the existing algorithms could not detect and eliminate the effect of the attackers, causing a large loss as illustrated in Fig. 5. It is interesting to notice that the performance of the existing attacker detection algorithms, i.e., FoolsGold and GeoMed, turn out to be worse than the baseline FedAvg algorithm. This is mainly caused due to incorrect elimination of the normal clients (detected wrongly as attackers) from the aggregation thus indirectly strengthening the collusion attack. This is evident in the case of the FoolsGold algorithm for the MNIST dataset where a steep jump in the loss appears after initial rounds of training.

In Fig. 5, the proposed algorithms show comparable loss to the existing algorithms for the initial rounds, but the loss decreases sharply afterward, which can indeed be verified by the earliest round of detection (ED) reported in Table II.



(a) Training Loss for MNIST (b) Training Loss for FMNIST

Fig. 4: Training loss over FL rounds for $A-5$ attack.

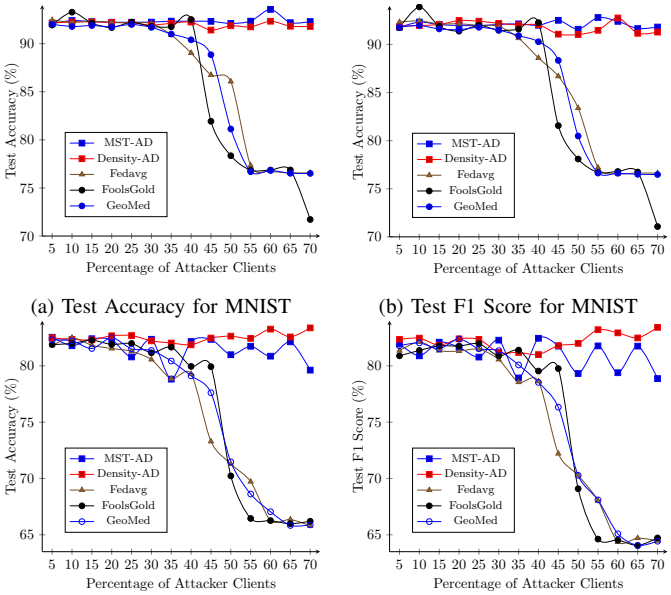


(a) Training Loss for MNIST (b) Training Loss for FMNIST

Fig. 5: Training loss over FL rounds for $A-70$ attack.

2) Impact of colluding attackers on test accuracy: Next, we report the results on test data, for both datasets, with a varying number of attackers in Fig. 6. It is clear that the proposed algorithms can maintain consistent performance with a larger number of attackers, however, the existing algorithms employing central measures like mean and median suffer from performance loss especially when the attackers overwhelm the normal clients. This can be attributed to the compared algorithms incorrectly classifying normal clients as attackers and excluding them from the aggregation process, thereby degrading the classification accuracy.

Similar observations can be made from the F1 Score for the algorithms reported in parts (b) and (d) of Fig. 6. Since the proposed algorithms are able to detect the full set of attackers, the obtained F1 Score values do not show much drop even when the number of attackers is more than 50%.



(a) Test Accuracy for MNIST (b) Test F1 Score for MNIST

(c) Test Accuracy for FMNIST (d) Test F1 Score for FMNIST

Fig. 6: Accuracy and F1 Score results under varying attack scenarios.

3) Analyzing ASR: The ASR and the earliest detection round (ED) of all the attackers are presented in Tables I and II for MNIST and FMNIST datasets, respectively. It is easy to observe that the proposed algorithms are able to maintain a lower ASR even when the proportion of attackers rises. As the proposed algorithms are able to eliminate the effects of the colluding workers, the successful number of attacks remains lower while the number of samples targeted by the attackers' increases, thus yielding a decrease in ASR. This is in contrast with the existing algorithms yielding a higher ASR following an increase in the number of colluding attackers. Among the considered existing algorithms, FoolsGold can successfully detect all the attackers only when their percentage is low however it fails in the majority of the cases and thus most entries in ED are marked by $*$. The proposed algorithms, on the other hand, are able to consistently identify the full set of attackers which is also reflected by their ASR.

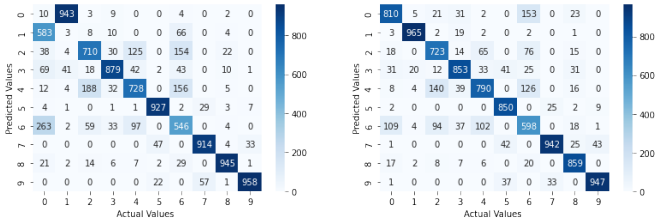
4) Confusion matrices: Finally, we report the confusion matrices for FoolsGold and Density-AD in case of overwhelming attackers (i.e., for $A-70$) in Fig. 7. We can clearly see that FoolsGold could not correctly classify the images of flipped labels (i.e., 0 and 1), whereas the proposed algorithms do not show any such confusion between these labels.

TABLE I: ASR and ED with varying attack scenarios for MNIST. We do not report ED for FedAvg and GeoMed algorithms as they do not focus on detection. [**ASR** – lower is better. **ED** – $*$ means the algorithm could not detect full set of attackers up to 30 rounds]

Atk	MST-AD		Density-AD		FoolsGold		FedAvg	GeoMed
	ASR	ED	ASR	ED	ASR	ED	ASR	ASR
$A-10$	0%	11	0%	10	0%	$*$	0%	0%
$A-15$	0%	9	0%	20	0%	20	0.3%	0%
$A-20$	0%	9	0%	15	0%	18	0.25%	0%
$A-25$	0%	10	0%	19	0%	17	3%	0.2%
$A-30$	0%	11	0%	14	0%	25	1.67%	0%
$A-35$	0%	9	0%	15	0%	24	13%	0.86%
$A-40$	0%	10	0%	16	0%	$*$	34%	1.5%
$A-45$	0%	11	0%	17	2.5%	$*$	19.2%	28.3%
$A-50$	0%	3	0%	18	3.4%	$*$	25.1%	25%
$A-55$	0%	4	0%	20	36.7%	$*$	60.18%	42.7%
$A-60$	0%	18	0%	15	77.4%	$*$	100%	74.41%
$A-65$	0%	11	0%	18	88%	$*$	99.7%	78.5%
$A-70$	0%	11	0%	9	100%	$*$	96.39%	94.9%

TABLE II: ASR and ED with varying attack scenarios for FMNIST. We do not report ED for FedAvg and GeoMed algorithms as they do not focus on detection. [**ASR** – lower is better. **ED** – $*$ means the algorithm could not detect full set of attackers upto 30 rounds]

Atk	MST-AD		Density-AD		FoolsGold		FedAvg	GeoMed
	ASR	ED	ASR	ED	ASR	ED	ASR	ASR
$A-10$	3%	3	3.5%	23	4.52%	$*$	2.5%	2.5%
$A-15$	2.67%	7	2%	14	1.64%	$*$	3%	2%
$A-20$	1.5%	3	1%	24	1.52%	22	3%	3.25%
$A-25$	1.8%	29	1.4%	16	1.4%	$*$	6.6%	1.4%
$A-30$	1%	18	1.67%	17	0.84%	$*$	2.3%	2.3 %
$A-35$	0.85%	3	1.08%	21	1.16%	$*$	19.8%	1.14%
$A-40$	0.5%	12	1.63%	22	6%	$*$	8.5%	4.25%
$A-45$	0.67%	3	0.88%	20	3.78%	$*$	82.32%	22.44%
$A-50$	1.1%	3	0.81%	21	43.2%	$*$	82.5%	47.45%
$A-55$	0.81%	2	0.6%	15	71.1%	$*$	71.27%	51.9%
$A-60$	1.83%	28	0.54%	16	66.15%	$*$	67.9%	62.5%
$A-65$	0.77%	2	0.58%	19	70.5%	$*$	70.15%	74%
$A-70$	1%	11	0.57%	18	76.3%	$*$	71.65%	76.25%



(a) FoolsGold (b) Density-AD

Fig. 7: Confusion matrices for FoolsGold and Density-AD algorithms in case of $A-70$ on FMNIST dataset.

V. Conclusion

In this paper, we attempted to address a critical problem of FL framework which is the presence of colluding attackers. Since the attackers can harm the global model severely, their detection is of utmost need for real deployment of the FL. We proposed two graph-based algorithms, MST-AD and Density-AD, by leveraging gradients' correlation among the clients. By performing an extensive set of experiments, we validated that the proposed algorithms can maintain a low attack success rate even when the attackers overwhelm the normal clients.

Since the proposed algorithms rely on correlation, they may not be able to detect an adversary if the FL system does not have any other adversary to collude with, which we plan to relax in the future. In addition, the current versions of proposed algorithms are limited to label-flipping attacks only. We plan to scale our FL setup by including more types of attacks such as byzantine, backdoor, and multi-label flipping.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses*, 2020, pp. 301–316.

[3] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in *31st International Conference on Neural Information Processing Systems*, 2017, pp. 3520–3532.

[4] C. Xie, M. Chen, P.-Y. Chen, and B. Li, "Crfl: Certifiably robust federated learning against backdoor attacks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 372–11 382.

[5] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *International Conference on Neural Information Processing Systems*, 2017, pp. 118–128.

[6] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[7] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *IEEE Trans. on Signal Processing*, vol. 68, pp. 4583–4596, 2020.

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

[9] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

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

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

[12] D. Wu, M. Pan, Z. Xu, Y. Zhang, and Z. Han, "Towards efficient secure aggregation for model update in federated learning," in *IEEE Global Communications Conference (GlobeCom)*, 2020, pp. 1–6.

[13] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *European Symposium on Research in Computer Security*. Springer, 2020, pp. 480–501.

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

[15] P. Ranjan, F. Corò, A. Gupta, and S. K. Das, "Leveraging spanning tree to detect colluding attackers in federated learning," in *IEEE Conference on Computer Communications Workshops*, 2022, pp. 1–2.

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

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