IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) | 978-1-6654-9427-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOMWKSHPS57453.2023.10225922

Robust Federated Learning against Backdoor Attackers

Priyesh Ranjan[1], Ashish Gupta[1], Federico Corò[2], and Sajal K. Das[1]
[1]Missouri University of Science and Technology, Rolla, USA
[2]University of Perugia, Italy
pr8pf@mst.edu, ashish.gupta@mst.edu, federico.coro@unipg.it, sdas@mst.edu

***Abstract*—Federated learning is a privacy-preserving alternative for distributed learning with no involvement of data transfer. As the server does not have any control on clients' actions, some adversaries may participate in learning to introduce corruption into the underlying model. Backdoor attacker is one such adversary who injects a trigger pattern into the data to manipulate the model outcomes on a specific sub-task. This work aims to identify backdoor attackers and to mitigate their effects by isolating their weight updates. Leveraging the correlation between clients' gradients, we propose two graph theoretic algorithms to separate out attackers from the benign clients. Under a classification task, the experimental results show that our algorithms are effective and robust to the attackers who add backdoor trigger patterns at different location in targeted images. The results also evident that our algorithms are superior than existing methods especially when numbers of attackers are more than the normal clients.**

***Index Terms*—Federated learning, backdoor, robustness, targeted attackers,**

I. Introduction

Federated Learning (FL) [1] is an emerging area of distributed learning which decentralizes the collection and processing of data to local participants. By maintaining participants' control over the usual privacy sensitive data and incorporating anonymity in model training, FL bolsters collaborative learning across participants by ensuring data privacy. Such collaborative learning leads to an efficient data utilization along with user and data anonymity [2]. However, this anonymity comes at the cost of the participation of adversaries with the intent to corrupt the model. Due to the lack of restriction on client training as well as the decoupling of the need of data collection, the adversaries can collude and hamper model convergence. An individual adversary can easily circumvent traditional defenses and can force non-optimal convergence [3]. A large number of such attackers can even overwhelm a strong robust aggregation scheme.

The detection and isolation of adversaries remain elusive due to the lack of information about the training process at the clients' end. Training of the model on corrupted data shards [4] or manipulated labels [5], [6] are commonly employed by adversaries along with model replacement [7] to maximize the malicious effect on the model. In particular, there exists a form of attack allowing the adversary to influence the global model to achieve high accuracy on a backdoor sub-task while retaining the performance on other tasks [8]. Such attacks, commonly known as backdoor, provide additional challenges during detection as the poisoning can stay dormant and can be activated during testing by providing the unique trigger that was incorporated by the adversary during training.

This work is motivated by the need of a novel method for secure and robust aggregation against backdoor attacks, which can identify the entire set of adversaries and isolate them during the communication round to effectively eliminate their effect during the training. While there are works discussing FL defenses for targeted attacks [9], the extension to backdoor attacks still remains a challenge due to their minimal impact on the main FL task. Other works that involve techniques like encryption for model transfer during server-client communication as in [10] further require the number of adversaries to be in minority to allow additional deliberation. Algorithms employing central tendencies like pairwise distances [11] or divergence [12] also assume the number of adversaries to be in minority and thus are vulnerable to overwhelming attacks on the setup. Unlike this, our work does not assume the set of adversaries to be in minority and is equally robust in cases where the benign agents are overwhelmed by the adversaries. We highlight the contributions of the work as:

- By exploiting the core ideas of Maximum Spanning Tree (MST) and K-Densest graphs, we propose two attacker detection (AD) approaches, namely MST-AD and Density-AD, that identify and effectively isolate the weight updates obtained from backdoor adversaries before the aggregation during training.
- Our algorithms empower the server to identify and exclude the entire set of adversarial agents at every communication round even in cases where these adversaries overwhelm (70% of total clients) the normal clients.
- We empirically demonstrate the effectiveness of MST-AD and Density-AD under an image classification task using a benchmark dataset and compare the performance with the existing state-of-the-art aggregation algorithms.

The rest of the paper is organized as follows. Section II formally introduces the problem and describes the assumptions made by the proposed algorithms along with the threat model considered. Section III describes the two proposed algorithms while Section IV ascertains the performance of these algorithms as compared to the existing state-of-the-art algorithms. The work is concluded in Section V.

978-1-6654-9427-4/23/$31.00 ©2023 IEEE

II. Problem Description

This section presents FL background and describes the goal of our aggregation algorithms. We also describe the adversary's goal and the assumptions made about their capabilities.

A. Federated Learning

FL relies on the collaborative training of a server model by participants without any movement of individual data shards. Mathematically, the set of participants $\mathcal{C}$ where each client $c_i \in \mathcal{C}$ has access to data shard D_i, provide their own local weight update δ_i^t to the server. These weight updates are aggregated into updating the server model W^t at communication round t. The Stochastic Gradient Descent algorithm, employed in conventional FL [1] operates as presented in Algorithm 1, where w_i^t and δ_i^t are the weights and weight updates respectively for an individual client $c_i \in \mathcal{C}$ having access to data shard D_i. ∇f and ϵ are the loss function and the learning rate for an individual client. p_i is utilized by the server during aggregation and decides the relative weight of individual participants' update vectors.

Algorithm 1: Stochastic Gradient Descent in FL

1 **Initialization:** Server Model W^0
2 **for** $t \in [0, T]$ **do**
3 **for** $c_i \in \mathcal{C}$ **do**
4 $w_i^t \leftarrow W^t$
5 $\delta_i^t = w_i^t - \epsilon \nabla f(w_i^t, D_i)$
6 $W^{t+1} = W^t + \sum_{i=1}^{n} p_i \delta_i^t, \sum_i^n p_i = 1$
7 **return** W^T

However, this aggregation does not take into account the presence of adversaries attempting to provide non-convergent weight updates. In particular, during such a scenario, the equation in Line 6 of Algorithm 1 modifies to:

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

where δ_j^t is the update vectors provided by the set of adversaries at communication round t and $\sum_{i=1}^{n-m} p_i + \sum_{j=1}^{m} p_j = 1$. We denote the corresponding set of adversaries as $\mathcal{M}$ making the corresponding set of benign workers as $\mathcal{B} = \mathcal{C} \setminus \mathcal{M}$. The algorithms proposed in this work aim to identify the set $\mathcal{M}$ and exclude the corresponding weight updates δ_j^t from the training process by manually enforcing the values of $p_j = 0, \forall c_j \in \mathcal{M}$.

B. Threat Model

The adversary aims to maximize the global models' performance on the test set images containing the backdoor trigger. The model performance on the original FL task, however, is expected to remain high. The adversary achieves this by embedding a trigger pattern in select images and manipulating the corresponding labels on its local data shard. The aggregated server model yields the expected classification outcome while performing classifications task on the images without the embedded trigger pattern in the test set. However, by embedding a similar trigger in test set images, the adversary can easily manipulate the global model toward misclassification into the targeted label.

C. Attacker Capability and Assumptions

We assume a Non-Independent and Identical (IID) distribution of data among the participating clients as suggested in [8]. We also assume that the participating clients do not have any knowledge or control over the aggregation method at the server end. Further, no client including the adversary has access to the local model of the benign clients. The adversary, however, has access to the local model of the clients under them and can control all the aspects of their local training. Further, we assume the number of adversarial clients to be at least 2 to realize collusion between the adversaries, and we assume the presence of at least 2 benign clients to leverage their weight update correlations.

III. Proposed algorithms

This section proposes two algorithms, MST-AD and Density-AD, for detecting adversarial clients in FL. While MST-AD is faster, Density-AD gives better and consistent results. We first describe the underlying principle of weight update correlations leveraged by the two proposed algorithms followed by the corresponding graph formulation.

A. Weight Correlations

As the weight update vectors are provided to the server at every communication round by every participating client, the correlation between each pair of client is leveraged by our work to separate the set of adversaries from the set of benign clients. This correlation between any pair of clients i and j is calculated 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 $\forall c_i, c_j \in \mathcal{C}, i \neq j$. δ_i and δ_j are the weight update vectors of clients c_i and c_j with $\bar{\delta_i}$ and $\bar{\delta_j}$ being the corresponding mean values ($\bar{\delta} = \frac{\sum_1^d \delta}{d}$) and d is the length of the weight update vector. We use the notation r_{ij} to describe the correlation between c_i and c_j throughout the entirety of this work. Also, for the purpose of the algorithm, the value of $r_{ii} \forall c_i \in \mathcal{C}$ is kept 0.

In order to compare these weight correlation values between a pair of clients, we tabulate the weight correlations into a matrix form notation. We depict the correlation matrix as $\mathcal{A} \in \mathbb{R}^{n \times n}$ where $n = |\mathcal{C}|$ is the number of clients participating in the training process. The obtained matrix $\mathcal{A}$ is a symmetric matrix from the symmetric nature of correlations and has 0 as its diagonal entries.

Correlations among different client types: Based on the observation in [5] we assert that the weight correlation between different a pair of client update vectors depends on the client type. In particular, the correlation between two adversarial clients is higher due to providing weight updates targeted

towards the common adversarial goal. The weight updates from the benign agents, however, show a higher divergence among each other thereby leading to a lesser correlation value between a pair of benign agents. Further, the correlation between an adversarial client and a benign client has an even lesser correlation value resulting from the different objectives while model training. To this end, we incorporate the following inequality in our proposed mechanism:

$$r_{m-c} < r_{c-c} < r_{m-m} \tag{3}$$

where m is an adversarial client from the set of adversarial clients $\mathcal{M} \subset \mathcal{C}$ and c is a benign client from the set of benign clients $\mathcal{B} = \mathcal{M} \setminus \mathcal{C}$. It should be noted that although the inequality in Equation 3 may not hold true for every pair of clients at every communication round, it can still be used to identify the set of adversarial clients.

B. Graph Representation

We now leverage the correlation matrix $\mathcal{A}$ obtained from the weight correlations to form a fully connected graph $\mathcal{G}$ where the nodes represent the participating agents and the edges represent the corresponding correlation values between the agents. The graph thus obtained is a complete graph with $\binom{n}{2}$ edges corresponding to the upper triangle entries in $\mathcal{A}$. The symmetric nature of $\mathcal{A}$ makes the graph and undirected matrix and the null diagonal entries make the graph free from self-loops. The obtained graph is then exploited in Section III-C and Section III-D to identify the set of adversarial clients from the set of benign clients.

C. MST-AD Algorithm

This section uses the graph realization obtained in Section III-B to identify the set of attacker clients by formulating a maximum spanning tree (MST). We employ a form of Kruskal's algorithm to identify the edge $edge$ with the maximum edge weight and add it into the spanning tree $tree$ as long as it doesn't form a cycle on $tree$. Then the edge is discarded from the tree permanently and the next $edge$ is chosen. This process is repeated until the $n-1$ edges for the tree have been appended. Following this, the edge with the smallest edge weight is then discarded from $tree$ resulting in two different sub-trees. Afterwards, the sub-tree with the lower median edge weight is flagged as the one containing the adversaries and the clients corresponding to its nodes are removed from the aggregation for that communication round.

The complete separation of the adversarial and benign clients into the two sub-trees relies on the assertion made in Equation 3. The algorithm first adds the edges between the pairs of adversaries because their correlations are higher than other correlations. Then, it adds the edges between the pairs of benign clients, as the correlation between these pairs is higher than the correlation between an adversarial-benign pair. This leads to a separate benign sub-tree disconnected from the adversarial sub-tree. The last edge added is a low-weight edge between an adversary and a benign agent, upon removing this edge, the two sub-trees with adversarial and benign agents are separated and their median edge weights can be compared to separate the adversarial sub-tree from the benign one as the adversarial-adversarial edge weights are higher than the benign-benign edge weights as asserted in Equation 3.

Algorithm 2: 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** $|trees| \leq n-1$ **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($\cdot$) computes average weight of tree */
9 **if** *med_weight*($subT_1$) $>$ *med_weight*($subT_2$) **then**
10 $Atk \leftarrow subT_1$
11 **else**
12 $Atk \leftarrow subT_2$
13 **return** Atk

Algorithm 2 illustrates the MST-AD algorithm using the correlation matrix $\mathcal{A}$. The algorithm creates an empty list to store the edges in the MST and sorts the edges in decreasing order (Line 3). It then iterates through every edge and appends it to the tree as long as it does not form a cycle (Line 6). Afterward, the lowest weighted edge is removed from the tree, resulting in two separate sub-trees (Line 8). The sub-tree with the higher median edge weight is then identified as the one containing adversarial nodes, based on the assumption that the correlation between adversarial clients is higher (Line 10).

D. Density-AD Algorithm

This section leverages the utilization of graph density metric to identify the set of adversaries. In this case we define density as the average of all the edge weights in the graph. Mathematically, it is defined as:

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

where n is the number of nodes in the graph and r_{ij} is a corresponding entry in correlation matrix $\mathcal{A}$.

The algorithm uses Eq. 4 to calculate the density of the graph. The first node is thereafter removed and the density of the resulting graph is calculated. If the removal of the node increases the density of the resultant graph, we flag the node as a sparse node and remove it permanently from the graph. Otherwise, the node is kept and the subsequent node is considered and the process is repeated. This allows us to separate the set of adversaries from the set of benign clients as the benign clients will be sparse nodes in cases with overwhelming adversaries (due to lower adversary-benign edge weight) and vice-versa. Removing these sparse nodes creates a perfect separation between the set of adversarial and benign clients as sparse nodes and the remainder of the graph. Afterward, among the sparse nodes and the remaining graphs,

the set having a higher median edge weight is flagged as the set containing the adversary nodes.

Algorithm 3: Density-AD Algorithm

1 **Input:** Correlation matrix $\mathcal{A}$
2 **Output:** Set of attackers (Atk)
3 $sparse\_list \leftarrow \emptyset$
4 **while** $i = n$ *down to* 1 **do**
5 $\quad \mathcal{B} \leftarrow \mathcal{A} \setminus \mathcal{A}[i]$
6 $\quad$ **if** $density(\mathcal{B}) > density(\mathcal{A})$ **then**
7 $\quad\quad sparse\_list \leftarrow sparse\_list \cup \mathcal{A}[i]$
8 $\quad\quad \mathcal{A} \leftarrow \mathcal{A} \setminus \mathcal{A}[i]$
9 **if** $med\_weight(sparse\_list) > med\_weight(\mathcal{A})$ **then**
10 $\quad Atk \leftarrow sparse\_list$
11 **else**
12 $\quad Atk \leftarrow \mathcal{A}$
13 **return** Atk

The Density-AD algorithm is illustrated in Algorithm 3. The list of sparse nodes in the matrix is initialized and the nodes are iterated through one-by-one (Line 4). Then a corresponding node is removed from the graph (Line 5) and the density of the resultant graph is calculated (Line 6). If the density of the graph without the node is higher than the density of the graph with the node, the node is flagged as a sparse node (Line 7) and is permanently removed from the graph (Line 8). After iterating and repeating this process through the entire set of nodes in the graph, the median weights of edges in the sparse list and the remaining graph are compared (Line 10) and the list with higher median edge weight is flagged as the list containing the adversaries.

IV. Experimental Evaluation

This section provides the experimental verification of the proposed algorithms under a FL framework and analyzes the results obtained in identification and elimination of the adversarial clients. The comparison with existing state-of-the-art methods is also reported to ascertain the superiority of the proposed models on various evaluation metrics.

A. Experimental Setup

Considering an image classification task, we train a model through FL using a benchmark Fashion-MNIST (FMNIST) dataset [13] consisting of $60,000$ training and $10,000$ testing 28×28 greyscale images that divided equally into 10 classes [T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot]. Here, the task involves classification of the category of the apparel from the corresponding greyscale images. The dataset is divided into $n = 50$ disjoint data shards for the 50 simulated clients. This partitioning follows Dirichlet distribution with the parameter $\alpha = 0.9$ as mentioned in [8]. Each client performs training for 10 local epochs on its local data within every FL round of communication. We consider a total of 30 such communication rounds and report the results obtained at the end of the 30^{th} round using the metrics described in Section IV-C.

B. Backdoor Attack Scenario

For the purpose of backdoor, we introduce a trigger pattern in selected images and flip their corresponding target labels to "Bag". In principle, the knowledge of the triggered pattern in the global model stays dormant until the specific trigger is encountered during testing upon which, the model would classify the test sample to the class "Bag". In addition, the location of trigger not fixed in the poisoned images, which makes the problem even more challenging for the detection algorithm. Fig. 1 depicts the considered trigger pattern incorporated in training images (of the targeted class) of the adversaries' data shards. Besides that the trigger's size also varies across images.

Fig. 1: Illustration of trigger patterns being embedded by backdoor attackers before local training.

Additionally, we conduct experiments with varying number of adversaries ranging from 5% to 70% of the total number of clients. The number is not increased beyond 70% to ensure fair comparison across different test scenarios as increasing the the ratio of adversaries decreases the ratio of benign agents thereby reducing the performance even in cases with perfect detection of adversaries. Further, inspired by [5], we abbreviate the attack scenarios as $A{-}m$ where m is the percentage of adversarial clients to the total number of clients.

C. Evaluation Metrics

We split the evaluation metrics into two categories.

*1) **Quantifying success on attacker detection**:* We consider the following three metrics to report the success of our approaches on adversarial identification:

- **Attack Success Rate (ASR):** It is defined as the ratio of the number of test samples incorrectly classified into the adversarial label (i.e., "Bag") to the total number of test samples containing the backdoor trigger. In the interest of fair evaluation, we omit the number of samples already belonging to the target class from the ASR metric.
- **Earliest round of Detection (ED):** We define ED as the earliest round in which the corresponding set of algorithms identifies the entire set of adversaries. We do mention that the metric does not apply to algorithms employing averaging or some other transformation on the entire set of clients and hence is omitted for the corresponding algorithms.
- **False Positives (FP):** It is defined as the number of benign agents incorrectly flagged as adversaries in the round of earliest detection (ED). As an aggregation algorithm flagging the entire set of clients on the first round of training performs well on the ASR and ED metrics, the False Positives metric is introduced to separate such aggregation from true detection of adversarial clients.

TABLE I: ASR, ED and FP with varying attack scenarios for Backdoor Attacks with a moving trigger pattern (**ASR** – lower is better.)

Atk	MST-AD			Density-AD			FoolsGold			FedAvg	GeoMed
	ASR	E.D.	F.P.	ASR	E.D.	F.P.	ASR	E.D.	F.P.	ASR	ASR
A-10	0.95%	2	0	0.80%	2	0	0.97%	1	24	8.59%	2.16%
A-15	0.83%	2	0	0.74%	2	0	0.76%	1	27	13.75%	4.72%
A-20	0.80%	2	0	0.68%	3	0	0.94%	1	28	25.47%	7.71%
A-25	0.97%	3	0	0.58%	3	0	0.97%	1	28	27.19%	17.99%
A-30	0.84%	2	0	0.60%	3	0	0.86%	1	22	34.04%	24.78%
A-35	0.82%	2	0	0.62%	3	0	0.90%	1	12	40.28%	38.40%
A-40	0.85%	2	0	0.76%	3	0	0.95%	1	22	46.04%	48.76%
A-45	0.89%	5	0	0.81%	7	0	0.95%	1	18	47.57%	79.31%
A-50	0.20%	2	0	0.60%	6	16	0.83%	1	10	51.92%	88.80%
A-55	0.90%	5	0	0.76%	5	13	0.94%	1	13	58.48%	90.00%
A-60	1.11%	2	0	0.57%	3	1	0.93%	1	7	60.45%	90.00%
A-65	0.38%	2	0	0.61%	3	7	0.93%	1	11	74.05%	90.00%
A-70	0.60%	2	1	0.57%	3	6	0.70%	1	0	85.55%	90.00%

*2) **Measuring classification performance**:* These metrics compare the classification performance of the collaborative model obtained in FL.

- **Accuracy:** We define accuracy as the ratio of the number of correctly classified samples in the test set to the number of samples in the test set. We provide only the test accuracy values as adversaries training on corrupted data shards renders training accuracy an ineffective metric.
- **F1 Score:** F1 Score is described as the harmonic mean of precision and recall values obtained on the test set. Mathematically, it is defined as:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2}{2 + \frac{FN+FP}{TP}} \quad (5)$$

  where FN, FP, and TP are False Negatives, False Positives, and True Positives respectively.
- **Training Loss:** We provide the value of the loss function obtained at every communication round for the $A-5$ and $A-70$ attacks.
- **Confusion Matrix:** Confusion Matrices for the $A-5$ and $A-70$ attacks are provided for comparison. For brevity, we only provide comparative matrices for the most competitive algorithm next to the proposed algorithms.

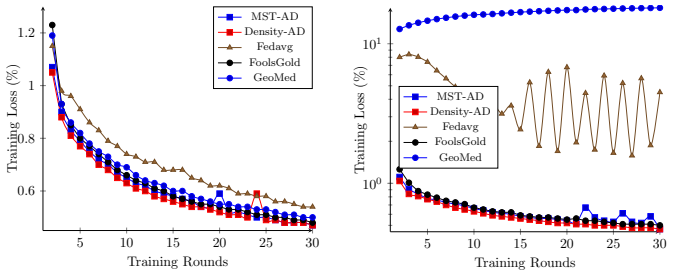
D. Results

We present the results in comparison with the existing competitive detection algorithms including FoolsGold [5] and GeoMed [14] to illustrate the superiority of the proposed algorithm. We also include the federated averaging (FedAvg) [1] to establish the baseline performance.

*1) **Robustness**:* Table I presents the detection metrics of Attack Success Rate, Early Detection, and False Positives. It can be seen that the algorithms relying on central measures like the FedAvg and the GeoMed fail to maintain a low Attack Success Rate when the number of adversaries increase. As these algorithms do not rely on detection and, as a consequence isolation, they are easily overwhelmed when the number of adversaries increase. While FoolsGold circumvents this issue by detection and isolation, it flags multiple benign agents as adversaries sacrificing model performance(seen in Section IV-D2) as a consequence. In contrast the proposed algorithms maintain a lower ASR comparable to FoolsGold and do not incorrectly flag benign agents as adversaries thus reducing the isolation's impact on performance.

*2) **Classification performance**:* Next, we report the classification performance results using the considering metrics.

• *Training Loss:* Figure 2 provides the training losses for the compared aggregation algorithms for the $A-5$ and $A-70$ attacks. We observe a convergence of the loss values for the algorithms for the $A-5$ attack. However, the loss values diverge quickly for the $A-70$ attack across the algorithms. While the FoolsGold, MST-AD and Density-AD algorithms show similar convergence and loss function values as the $A-5$ attack, the FedAvg algorithm shows non-convergence and large oscillation values of the loss function. Further, the GeoMed algorithm shows divergence away from the optimum loss value showcasing its poor performance when the percentage of adversaries increase.

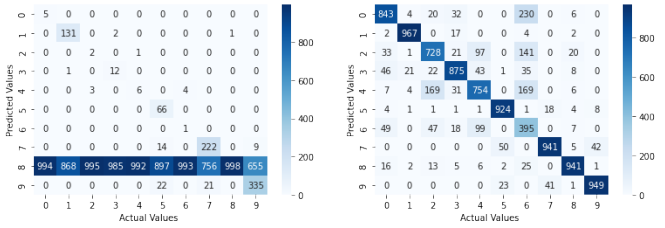


(a) Training Loss for $A-5$ attack (b) Training Loss for $A-70$ attack

Fig. 2: Training loss over 30 communication rounds.

• *Confusion Matrices:* We present the confusion matrices for FedAvg and the Density-AD algorithm in Figs 3(a) and 3(b), respectively. We can observe that the Density-AD algorithm provides better performance on the test set as compared to the FedAvg algorithm on the $A-70$ attack as the adversarial attempt at misclassification of the target label into Label '8' is successful. The proposed algorithms are able to identify and eliminate the adversaries thus resulting in high classification performance on the task.

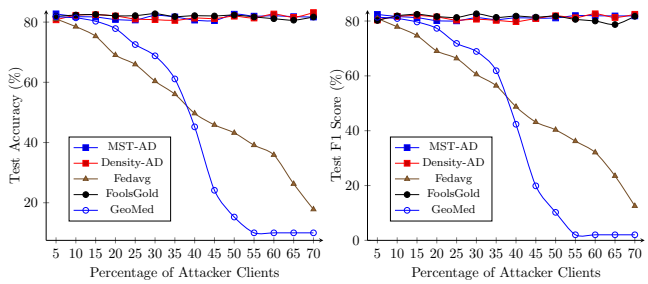
• *Accuracy and F1 Score:* We depict the Accuracy and F1 Score Metrics in Figure 4 for the proposed algorithms. We



(a) FedAvg (b) Density-AD

Fig. 3: Confusion matrices for FedAvg and Density-AD algorithms in case of $A-70$ on FMNIST dataset.

notice that while the proposed algorithms (and FoolsGold) are able to completely mitigate the impact of adversarial agents on the model performance, the algorithms employing central measures show a sharp decline in performance following the increase in the number of adversaries. We further see a slight decline in the performance of the FoolsGold aggregation as the number of adversaries increase due to incorrect flagging of benign agents as adversaries.



(a) Test Accuracy (b) Test F1 Score

Fig. 4: Accuracy and F1 Score results under varying attack scenarios for backdoor attacks for moving trigger pattern.

Comparison of MST-AD and Density-AD algorithms: From the obtained results, it can be established that the performance of the Density-AD algorithm exceeds the performance of the MST-AD algorithm in most cases. The E.D. and F.P. values of the Density-AD algorithm degrade when the number of adversaries are equal to the number of benign agents whereas MST-AD algorithm shows consistent performance in all cases. Further, MST-AD algorithm incurs lower computational complexity of $O(n\log(n))$ than Density-AD algorithm with $O(n^2)$, where n is the number of edges in the graph.

V. Conclusion

In this work, we addressed the presence of adversaries employing data poisoning on the FL setup with added backdoor trigger pattern. By leveraging the graph theoretic algorithms of MST-AD and Density-AD, we were able to successfully identify and isolate the set of adversaries thus mitigating the effects of the attack. The experimental evaluations against a varied set of metrics and the existing state-of-the-art, we were able to establish the superiority of the proposed algorithms even in cases with overwhelming number of adversaries. In future, we plan to incorporate cases with different adversarial objectives employed by the backdoor triggers. Additionally, the extension to byzantine attacks and adversaries employing a combination of label flipping and backdoor attacks will be considered as an area of improvement of this work.

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. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10 782–10 793, 2020.

[3] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] A. Gupta, T. Luo, M. V. Ngo, and S. K. Das, "Long-short history of gradients is all you need: Detecting malicious and unreliable clients in federated learning," in *European Symposium on Research in Computer Security*. Springer, 2022, pp. 445–465.

[10] L. Zhao, J. Jiang, B. Feng, Q. Wang, C. Shen, and Q. Li, "Sear: Secure and efficient aggregation for byzantine-robust federated learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3329–3342, 2021.

[11] S. Awan, B. Luo, and F. Li, "Contra: Defending against poisoning attacks in federated learning," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 455–475.

[12] Y. Mao, X. Yuan, X. Zhao, and S. Zhong, "Romoa: Robust model aggregation for the resistance of federated learning to model poisoning attacks," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 476–496.

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

[14] 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.