A Clustering-Based Scoring Mechanism for Malicious Model Detection in Federated Learning

Cem Caglayan
Computer Engineering Department
Bogazici University
Istanbul, Turkey
cem.caglayan1@boun.edu.tr

Arda Yurdakul
Computer Engineering Department
Bogazici University
Istanbul, Turkey
yurdakul@boun.edu.tr

***Abstract*—Federated learning is a distributed machine learning technique that aggregates every client model on a server to obtain a global model. However, some clients may harm the system by poisoning their model or data to make the global model irrelevant to its objective. This paper introduces an approach for the server to detect adversarial models by coordinate-based statistical comparison and eliminate them from the system when their participation rate is at most 40%. Realistic experiments that use non-independent and identically distributed (non-iid) datasets with different batch sizes have been carried out to show that the proposed method can still identify the malicious nodes successfully even if some of the clients learn slower than others or send quantized model weights due to energy limitations.**

***Index Terms*—Federated Learning, Distributed Learning, Clustering, Anomaly Detection, Distributed system sustainability, Coordinate-based Scoring**

I. Introduction

The rise of the Internet of Things (IoT) has led to the emergence and active use of billions of devices connected to the internet. Consequently, a massive amount of data is generated that can be used for producing actionable insights. Privacy is a major concern when data have to travel across several devices on the internet. The solution comes with federated learning where each device does its training locally with its data, then transmits its model to a server. The server aggregates the collected client models with specific algorithms such as federated average (FedAVG) [1] which averages the models. However, several approaches can be replaced as aggregation models [2]. Finally, the server transmits the aggregated model to each client so that it can continue training the aggregated model as its new model. This communication between the server and the clients continues until an acceptable global model accuracy is obtained for all participants of the learning system.

Due to privacy issues, the server never knows the intention of a client in federated learning. This is an opportunity for malicious devices to harm the system by sending poisoned models. They may poison their data to collapse the aggregated model by changing their labels or spoiling the training data. They may also send an entirely different or statistically poisoned model instead of a trained model. Hence, current research focuses on server-side methods to prevent model breakdown from malicious users. There are two different approaches to the detection of adversaries. The first one is to find malicious devices in the network [3][4][5] where the traffic activities of the devices are collected as training data to detect unusual traffic. However, anomaly detection in this area requires pretraining to discover the regular traffic flow. The second one is to detect adversarial models among the regular ones that train widespread deep neural networks (DNN) [6] [7] where the malicious devices can ruin global models with specific algorithms. The server uses filtering and classification methods to categorize the poisoned models in the detection system.

This paper proposes an adversarial model detection system in deep neural networks by combining statistical methods with clustering algorithms. Clustering algorithms are widely used in detecting anomalies in federated learning systems. This work differs from the existing studies in the sense that collected client models are analyzed iteratively and statistically to generate multidimensional data points for each client to be used in the clustering mechanism. The data structure used in clustering has shown to be robust in extracting malicious client models when the adversary rate is not greater than 40%. The performance of the method is also evaluated with slow learning devices and devices that can send only quantized weights due to energy constraints. Thus, the proposed detection mechanism allows some devices to be different from the rest of the clients, which is not analyzed deeply in the literature. The test cases are divergent due to the heterogeneity of the clients. A framework has also been developed for this purpose to generate various attacks from the attack server.

The rest of the paper is organized as follows. The next section presents related studies and compares them with this work. It also includes the background to introduce the attack types. In Section III, the proposed method is explained. The framework and the experiments that are carried out on this framework are presented in Section IV. The final section concludes the paper.

II. Related Work

Recent research for adversary detection on federated learning systems can be studied with respect to network activities and model parameters. The former targets the detection of malicious activities on the network. The solutions in this

category mostly employ autoencoders, where encoders compress the input and decoders try to reconstruct it with a model. Autoencoders have to be trained with benign traffic data in order to discover abnormal network activity. When traffic is normal, reconstruction error is low. Abnormal traffic causes a high reconstruction error. Based on this fact, various studies exist with unique threshold techniques. In [3], the auto-encoders are selected as the defense mechanism for the unsupervised models where the clients can access only their benign traffic data. The mean square error (MSE) is used for calculating the threshold. Binary classification is carried out with multi-layer perceptrons in the supervised model, where the clients can access all their labeled data. In [8] and [9], threshold is computed as the sum of the mean and weighted standard deviation of MSE. In [10], Gated Recurrent Units (GRUs) are used to find maliciousness in traffic flow by giving "occurrence probability" to the structured network packets.

In the second category, the detection systems are implemented to find an anomaly in the deep network itself that is learned by the attendants of the federated learning system. The attackers infect the network with their local models in the federated learning mechanism. Thus, the detection system needs to check the validity of every client model according to its parameters/gradients/weights. In [6], the dimension of gradients is firstly reduced with principal component analysis(PCA). Then, euclidean distances between each client are calculated to eliminate the malicious clients with a threshold. Finally, the k-means algorithm clusters the gradients to measure cosine similarities for finding optimal gradients. Another study that uses cosine similarity to detect adversarial weight updates calculates the possible update direction calculated for the bad clients to eliminate them from the system [7]. Their solution relies on the fact that the benign clients have to exceed malicious ones as is the case for our work. In [11], autoencoders that uses MSE as the reconstruction error are utilized. Their solution requires all clients to be benign during the training phase of the autoencoder so that the decoder will be able to identify the adversaries in the upcoming rounds. This approach surely fails in case that malicious clients exist during the initial training phases of the federated learning system. There also exist several studies that use model classification with generative adversarial networks (GAN)[12], and k-means clustering on kernel principal component analysis (KPCA) [13].

Clustering methods are widely used in anomaly detection. In [13], the k-means algorithm is adopted in the adversary detection system as an additional layer to KPCA to compare the results. Another detection system [14] uses the same algorithm to cluster pair-wise L2-Norm distances within two clusters to detect malicious models. Similarly, in [15], the authors use a mechanism to cluster the masked features of the DNN model learned collaboratively. It also employs an error-rate tolerance metric on clusters to distinguish the adversary.

Our work is also in the second category where adversary model detection is aimed. Similar to existing studies in the literature, the detection mechanism is based on non-complex statistical scoring and threshold computation with the utilization of model weights to detect the malicious models. However, in our approach, several scoring levels are generated, feeding the k-means clustering algorithm to reduce the number of data points and analyze the models deeply and iteratively. Similar to the proposal in [14], the cluster with the higher number of attendants is considered benign as long as the distance between the clusters exceeds the threshold. In addition, clients may have different computation resources and power budgets. They can be subject to non-identical data sets. Also, as explained in Section III, there is no need to pre-train our detection system, which must be done in auto-encoders.

A. Attacks

In the literature, model and data poisoning attacks are targeted. In model poisoning attacks, the attacker aims to destruct the global model by adjusting the trained parameters. The gradient factor attack (Eq. 1) is an example [3] where the attacker transmits an adjusted model to make the global model's update -1 instead of 1. The malicious clients achieve this aim by multiplying their models by a hyperparameter α which is calculated by the count of total clients:

$$\frac{1}{C}.(C - M + \alpha.M) = -1 \tag{1}$$

Here, C is the number of clients and M is the malicious client count. Model canceling [3] is another example to poison the global model. The objective is to make the global model parameter equal to zero instead of one:

$$C - M + \alpha.M = 0 \tag{2}$$

To control the hyperparameter calculation method in divergent scenarios, there exists a specialized attack server that knows C and organizes M of them as adversaries by informing them. In addition to these hyperparameters, the attack has a poisoning rate parameter P_{rate} to decrease the rate of attacker identification in the detection stage. The model is multiplied with α and P_{rate} as the poisoned model in Eq. 3.

$$Model_{poisoned} = Model \cdot P_{rate} \cdot \alpha \tag{3}$$

Data poisoning attack is realized by malicious codes running on the clients. In label flipping attack, the malicious code flips the labels of training data to make the local model useless [3, 7, 12, 13]. For example in [7], the attack is realized by turning every data label to "0". Training data can also be remotely changed by injecting undesired patterns [6].

In this paper, label flipping and model poisoning attacks are targeted.

B. Quantized Models and Slow Learners

Most of the IoT end devices are resource-constrained. Especially, power consumption is a major concern for nodes running on batteries. Since federated learning requires multiple data transfers between the clients and the server, model quantization exists as a solution for these devices. Some studies utilize quantized parameters during training [16, 17]. Another

study realizes anomaly detection with gradient compression [18]. Slow learners represent the variety of clients that the federated learning structure contains. They may have smaller batch-size or local batch training. Federated rounds can be reduced for some devices, which have to use their limited power budgets effectively. Hence, they may occasionally attend in federated rounds [16].

In this paper, a post-training quantization technique is used. Float16 quantization transforms the variables from float32 to float16 [19]. Moreover, maliciousness analysis is also carried out on federated learning systems including quantized nodes, slow learners as well as full capacity devices.

III. Method

Let C be the set of clients in a federated learning system. At any time, there can be some malicious clients. Let M represent the set of malicious clients at a specific round of aggregation: $|M| < |C|/2$. The server sends the same model to all clients. Let $\{w_1, w_2, \ldots, w_i, \ldots, w_N\}$ denote the parameters of the model to be trained. At a specific round of training, the model at each client $c \in C$ can be viewed as $\{w_{1,c}, w_{2,c}, \ldots, w_{i,c}, \ldots, w_{N,c}\}$ (Alg. 1 lines 1-5). Since the server computes each model weight by aggregating the parameters from all clients, w_i can be regarded as a statistical variable with a mean μ_i and variance σ_i (Alg. 1 lines 6-7). When there are no malicious nodes in the system, the variance will be small. The existence of malicious nodes will cause a greater variance. Hence, the server should aggregate only the client parameters that exist in some range $[t_i^-, t_i^+]$ around the mean (Alg. 1 lines 8-9):

$$t_i^- = \mu_i - \theta.\sigma_i \tag{4}$$

$$t_i^+ = \mu_i + \theta.\sigma_i \tag{5}$$

Since the server is unaware of the malicious nodes at the aggregation time, it cannot specify an exact value of θ. If it is not correctly set, malicious nodes can enter the aggregation or too many benign nodes can be left outside. Hence, the server has to scan the client parameters for a set of different values for $\theta = [\theta_{\min}, \theta_{\max}]$. Empirical studies show that $\theta_{\min}$ should be much less than σ_i and $\theta_{\max}$ should be at least one σ_i ahead of $\theta_{\min}$.

Let $x_{i,c}$ be the binary variable that is 1 if and only if $w_{i,c}$ is in the range: $t_i^- \leq w_{i,c} \leq t_i^+$ (Alg. 1 lines 11-12). Then the amount of parameters of a client that can take place in the aggregation of a specific round can be computed as $V_c = \sum_{i=1}^{N} x_{i,c}$ (Alg. 1 line 13). Based on this fact, a score can be calculated for each client as follows:

$$S_c = \frac{V_c}{N} * 100 = \frac{\sum_{i=1}^{N} x_{i,c}}{N} * 100 \tag{6}$$

Eq. 6 assigns a score in $[0 - 100]$ range to each client in the system for every value of θ (Alg. 1 line 14). Assume that there are L distinct values for θ. Then there will be L scores for each client corresponding to all values of θ. The scores of a client can be collectively represented as L-dimensional data point with θ_l increasing from the first entry to the last entry (Alg. 2 lines 1-5):

$$\vec{\mathbf{S}}_c = \{S_{c,1}, S_{c,2}, \ldots, S_{c,l}, \ldots, S_{c,L}\} \tag{7}$$

In this way, there will be $|C|$ data points fed to a k-means clustering algorithm for two clusters of uneven size so that data points corresponding to malicious and benign clients can be placed into two disjoint clusters (Alg. 2 line 6). Let $\vec{\mathbf{O}}_1$ and $\vec{\mathbf{O}}_2$ be the centroids of these clusters. The proposed method assumes that the majority of clients are benign. When some malicious clients exist in the federated learning system, the distance between the centroids of each cluster, $|\vec{\mathbf{O}}_1 - \vec{\mathbf{O}}_2|$, will become large. If the system consists of all benign nodes, then $|\vec{\mathbf{O}}_1 - \vec{\mathbf{O}}_2|$ will be small. To obtain a measure independent of L, the distance between centroids of the clusters should be normalized by L (Alg. 2 line 7):

$$d_{O1,O2} = \frac{|\vec{\mathbf{O}}_1 - \vec{\mathbf{O}}_2|}{L} \tag{8}$$

The server has to check $d_{O1,O2}$ against a threshold γ to decide whether to use both clusters or only the larger one in the aggregation. Note that $d_{O1,O2}$ can have a value in $[0, 100]$ range due to the definition of the client score in Eq. 6. Hence, setting γ to a low value around 10 usually provides the necessary filtering. When γ is set to a very low value such as 5, some of the benign nodes can be left from the aggregation if there are no malicious nodes. When γ is set to a very high value such as 60, malicious client weights appear in the aggregation if they exist in the system.

Clustering can be extended to handle various types of nodes in the system. In almost every federated learning system, the server knows the learning types of the nodes: Some nodes can learn slower than others. Some nodes can provide quantized weights due to their characteristics. Hence, the server may prefer to have multiple clusters in the system. Still, in that case, the proposed method generates the clusters. It is up to the program running on the server to select the clusters in the aggregation by setting up separate γ values for each case.

IV. Experiments

A. Experimental Setup

A framework shown in Figure 1 is developed for experimental demonstration of the proposed method. It can work in real-life cases with a slight change in the communication system. It operates unanimously with TensorFlow Keras[20] framework on Tensorflow dataset[21]. A single batch script is developed to start each client by different processes. Thus, they are working with different terminals and Python processes. All Python objects communicate with the pipe communication system, including the weight parameter communication. Initially, the clients subscribe to the federated learning network. The federated server controls the clients to take the local model or transmit the global model. Thus, a single client is not connected to the other clients directly. The attack control server and federated server also run independently. In this

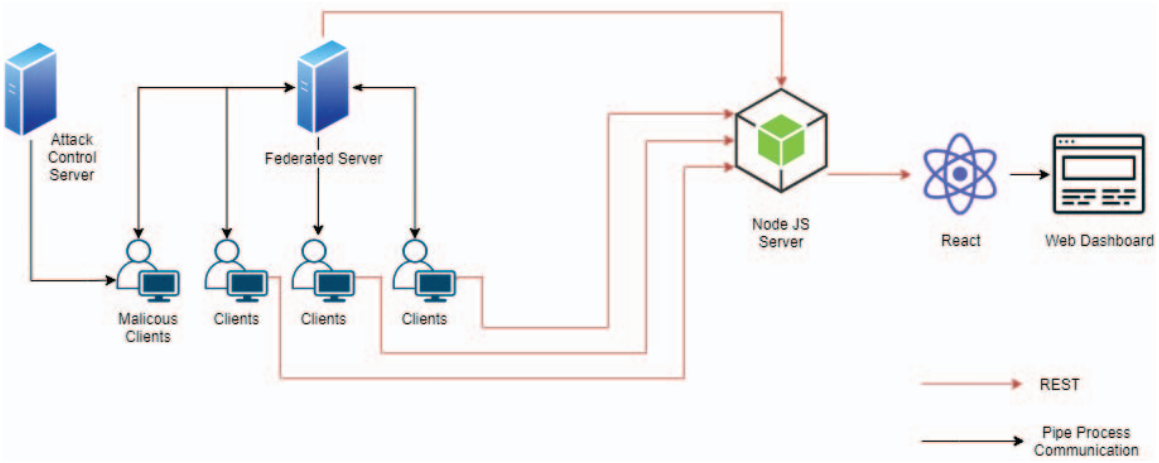


Fig. 1: The Experimental Setup

Algorithm 1 Score Calculation

Input: CModels, theta
Output: Score of each Model

$cmLength \leftarrow CModels.length$
$ScalingParameter \leftarrow 1/cmLength$
for $i \leftarrow 0$ ... $cmLength$ **do**
    $tempModels[i] \leftarrow CModels[i] * ScalingParameter$
end for
$meanModel \leftarrow mean(tempModels)$
$stdModel \leftarrow std(CModels)$

$tHigh \leftarrow meanModel + theta * stdModel$
$tLow \leftarrow meanModel - theta * stdModel$

for $i \leftarrow 0$ ... $cmLength$ **do**
    $lowScore \leftarrow isGreater(CModels[i], tLow)$
    $highScore \leftarrow isLower(CModels[i], tHigh)$
    $existScore \leftarrow inrange(lowScore, highScore)$
    $Score \leftarrow count(existScore)/W * 100$
end for

Algorithm 2 Malicous Model Detection

Input: CModels, thetaMin, thetaMax, thetaStep, threshold
Output: Benign CModels

for $i \leftarrow thetaMin$ ... $thetaMax$ **do**
    $tempScores \leftarrow calcScore(CModels, i)$
    $Scores \leftarrow assignScore(tempScores, i)$
    $i \leftarrow i + thetaStep$
end for

$clusters \leftarrow calcKMeans(Scores)$
$dist \leftarrow distance(clusters)$

if $dist \geq threshold$ **then**
    $CModels \leftarrow deleteMaliciousModels(CModels)$
end if

setup, neither the server nor the clients can access other clients' data.

Federated learning works better with identical and independently distributed (IID) data [22]. However, in real-world applications, the data is generally non-IID [23]. Since the clients of this paper are IoT devices, non-IID datasets are generated and used. In the experiments, data is thoroughly shuffled to make each client unique in the structure. Every object has its unique port number for the primary identification of the clients. In addition to the communication, the port number is also used as a randomness seed of a client. The randomness mechanism uses the seed to generate a unique and independent dataset from the selected Tensorflow dataset for each client. This feature also enables the dataset of each test case to be the same since the seed is unique for every client.

The federated server controls the federated learning process between clients The server has a subscribers list that consists of client addresses. If a client wants to connect to the system, the server also accepts an additional parameter: the frequency of a model's training. Suppose a client declares a slower communication frequency than the server's demand. In that case, the server identifies that client as the *slow learner* and skips collecting the parameters from that client at some rounds. The differentiation of slow learners among other clients is detailed in the experiments. The server's duty includes the anomaly detection system for the collected client models via the proposed method. In the final stage, the server aggregates the remaining models from the detection system and transmits the aggregated model to each client for the next round.

Clients do local training with the Stochastic Gradient Descent (SGD). They send their weights directly to the federated server whenever they are triggered by the server. Slow learners

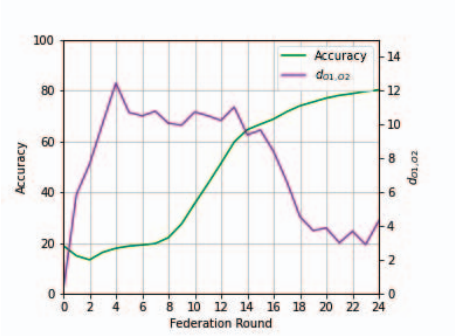


Fig. 2: Cluster Difference $d_{O1,O2}$ and Accuracy during Federated Learning / $|C| = 10, |M| = 0$

are triggered less frequently than other clients. Some clients may quantize their model with IEEE standard float16 format to send the quantized model. Some clients may be problematic. These devices do not try to harm the system; however, they train problematically rather than the standard client.

The experimental setup also includes a particular server to organize the attacks. The attack control server does not directly transmit data to the federated learning model. When a client wants to be malicious, or if malware takes the device's control, the client can generate a connection to this server and is controlled by the Attack Control Server. The server adjusts the attacking parameters and controls the attack type. The attacking parameters depend on the number of malicious ($|M|$) and total clients ($|C|$) to collapse the global model.

The system is setup with ten clients. Though the aggregation method is programmable in the framework, federated average [1] is used in the experiments since harmful models are eliminated from the system with the proposed method. The scanning parameter θ is defined as follows: $\theta_{min} = 0.7, \theta_{max} = 1.6, \theta_{step} = 0.1$. Hence, ten data points constitute a score for each client at each round. The dataset consists of hand-written numbers, which is the regular MNIST[24] dataset, and the model is a Convolutional neural network(CNN) model that includes 2 Convolution Layers, 2 Max-Pooling Layers, Flatten, and Softmax activation.

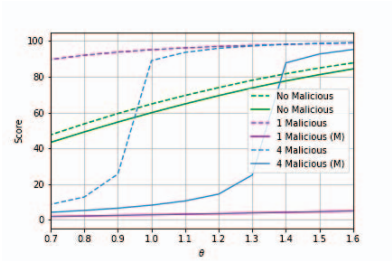
B. Experiment Results

Figure 2 shows the accuracy versus $d_{O1,O2}$ in a federated learning network where all clients are benign, learn at full rate, and communicate at full word length. Learning is done in 25 rounds. In each round, each client takes four local batches(lb). Each local batch represents four batches of data; thus, each round means 16 batches. Batch size is 64. Dataset of clients is not non-iid. The accuracy is generally increased in each round. Additionally, $d_{O1,O2}$ decreases while the accuracy starts to increase. This situation can be interpreted as lower accuracy means a higher possibility for the malicious devices to affect the system because of non-adjacent clients.

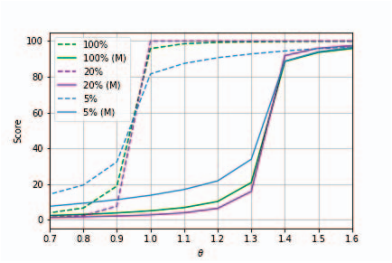
To test our malicious client detection system, some clients are transformed from benign to malicious. The gradient factor attack with 100% poisoning rate is used by the malicious clients. The scores of cluster centroids, namely $\mathbf{O_1}$ and $\mathbf{O_2}$, for each θ are shown in Figure 3a. In this figure, dashed lines represent the centroid of scores for the cluster with more clients while solid lines are used for showing the same information for the cluster with fewer clients. Recall that the smaller size cluster may imply the set of malicious nodes if $d_{O1,O2} > \gamma$. When there is one malicious client, the centroids of two clusters are far away from each other at all values of θ as shown by dashed and solid purple lines. When the number of malicious clients increases to four, the centroids of two clusters may come closer at some values of θ as seen with blue lines. However, the departed regions cause the overall distance to be $d_{O1,O2} = 36.92$ which is a value greater than $\gamma = 15$. So, in both cases, the cluster of malicious clients can be identified and erased before the aggregation. In the same figure, green lines demonstrate a distance of 4.31 between two clusters. Since this value is smaller than γ, the server takes the clients of both clusters into aggregation.

The effect of different poisoning rates is shown in Figure 3b. In this scenario, four malicious clients are taken into account with different poisoning rates. It can be seen that even 5% poisoning can be detected by the proposed method since $d_{O1,O2} = 31.22$ is still considerably higher than the threshold. In Figure 3c, the effect of malicious clients on $d_{O1,O2}$ at different rounds can be viewed. The test has been made with four malicious clients, which send their poisoned model at the end of four local batches. As can be seen, $d_{O1,O2}$ is close to the specified threshold γ at the first round with a 5% poisoning rate. Besides, the clustering algorithm classifies a benign client as malicious in addition to the other malicious ones at the tenth round at this poisoning rate. Thus, the model of this benign client will also be deleted with malicious models. This wrong classification may be neglected since the poisoning rate is low. This problem is not present when the attack is realized at a round where the global model accuracy is high even though the poisoning rate is small. For higher poisoning rates, it can be correctly detected at all rounds.

In the next test set, different types of clients are studied. The system is implemented with one slow learner and one problematic device in addition to eight regular devices. The tests are made with four malicious devices with different poisoning rates. Since the server is aware of slow learners due to the registration parameters, it prepares the detection system for three clusters: regular, slow learners, and others. Note that the server is aware of neither malicious nor problematic devices. The results can be seen in Figure 4a. In this figure, dotted lines represent the third cluster. The dashed and solid lines are used as before. When the malicious clients attack the server with 100% poisoning rate, the detection system can put every benign client in the same cluster as expected. The centroid score is very high. However, the other two clusters contain only malicious clients. The distance parameters tell the server to discard both of them. If malicious clients lower the poisoning rate to 20%, the benign clients can be successfully separated into two clusters for slow learners and regular nodes.



(a) 100% poisoning rate. $d_{O1,O2}$ is 4.31 for $|M| = 0$, 92.47 for $|M| = 1$, 36.92 for $|M| = 4$

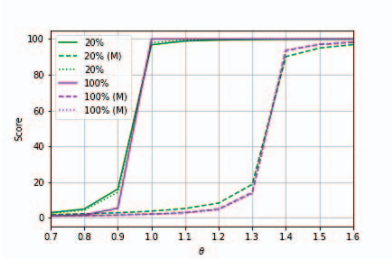


(b) various poisoning rates with $|M| = 4$. $d_{O1,O2}$ is 39.22 at 100%, 39.24 at 20%, 31.22 at 5% poisoning rates

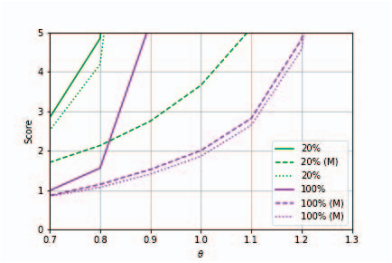
Round	Poisoning Rate		
	100%	***25%***	***5%***
$1^{st} Round$	60.94	51.87	15.73
$10^{th} Round$	38.97	38.32	32.10
$20^{th} Round$	39.13	39.41	34.73

(c) $d_{O1,O2}$ at different rounds and poisoning rates when $|M| = 4$

Fig. 3: Comparison of Clusters' Centroids when all nodes are regular and model poisoning attack is present



(a) $|C| = 10, |M| = 4$ / 1 Slow Learner, 1 Problematic Client

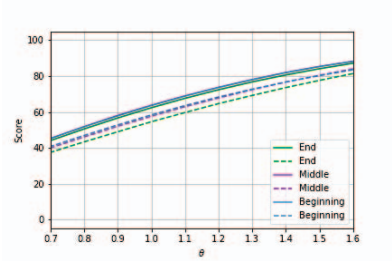


(b) Zoomed in Figure 4a

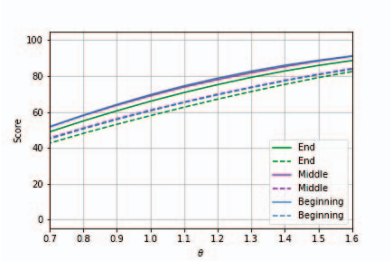
Clusters	Poisoning Rate	
	100%	***20%***
Solid (O1)	$\|M\| = 0$	$\|M\| = 0$
Dashed (O2)	$\|M\| > 0$	$\|M\| > 0$
Dotted (O3)	$\|M\| > 0$	$\|M\| = 0$

(c) Malicious Count of the Clusters in Figure 4a, 4b

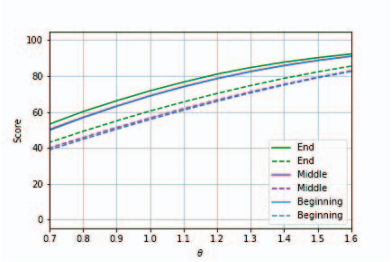
Fig. 4: Comparison of Clusters' Centroids; Slow and problematic learners / $d_{O1,O2}$ is 39.42, $d_{O1,O3}$ is 0.03, $d_{O2,O3}$ is 39.38 at 20% Poison Rate / $d_{O1,O2}$ is 39.17, $d_{O1,O3}$ is 39.28, $d_{O2,O3}$ is 0.11 at 100% Poison Rate



(a) 1 Slow learner / $d_{O1,O2}$ is 7.16 for "End", 5.52 for "Middle", 5.06 for "Beginning"



(b) 2 Slow learner / $d_{O1,O2}$ is 7.34 for "End", 6.02 for "Middle", 8.43 for "Beginning"



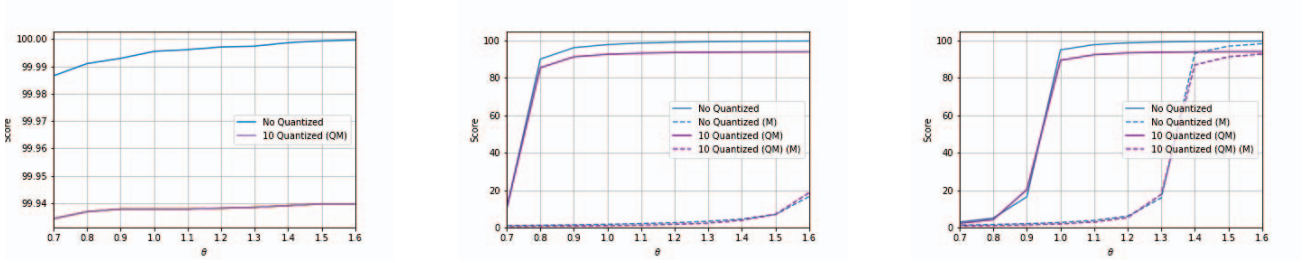
(c) 4 Slow learner /$d_{O1,O2}$ is 9.93 for "End", 11.02 for "Middle", 11.54 for "Beginning"

Fig. 5: Comparison of Clusters' Centroids; Slow and problematic learners; Beginning: Near to the beginning, Middle: middle of the process, End: final result / 24 round with 4 LB, accuracy is around 75%-80% percent at the end of each test / $|M| = 0, |C| = 10$

In this case, all malicious clients are listed in a single cluster. The distance parameters tell the server to discard this cluster. The zoomed figure shows the clustering in Figure 4b; and the content of the clusters is listed in Figure 4c.

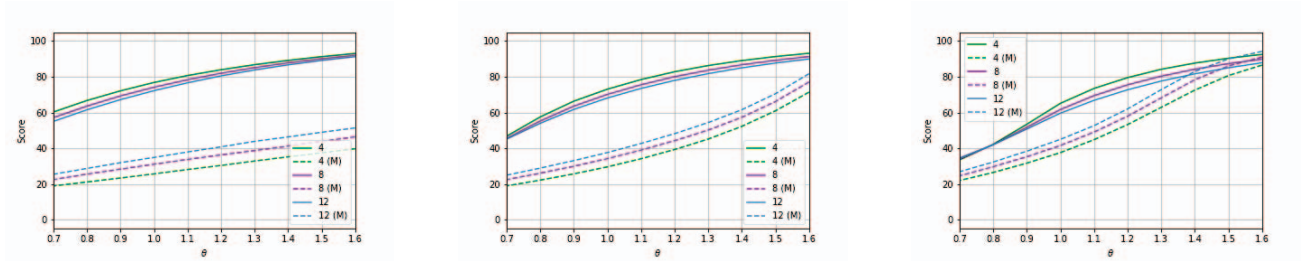
When the system does not contain any malicious nodes, the score between the benign and slow learners is very small. The score difference between various slow learner counts can be seen in Figure 5a, 5b, 5c. The results are collected from 3 different random points according to the federation round time(near to beginning, middle, end). The slow learners have a low impact on the scoring scheme. Slow learners can be dispatched into different clusters. However, this is not a problem for the federated learning system as long as the distance between clusters is lower than the threshold γ.

The final set of experiments for model poisoning is done with ten clients which send their model parameters after quantizing with float16. The accuracy is between 75-80% in every test. Malicious clients attack with 100% poisoning rate. Ten quantized models with only one malicious model have a slightly lower score than ten regular benign ones as shown in



(a) $|M| = 1$ (b) $|M| = 3$ (c) $|M| = 4$

Fig. 6: Comparison of Clusters' Centroids with zero quantized models(No Quantized) and ten quantized models (10 Quantized including malicious (QM))

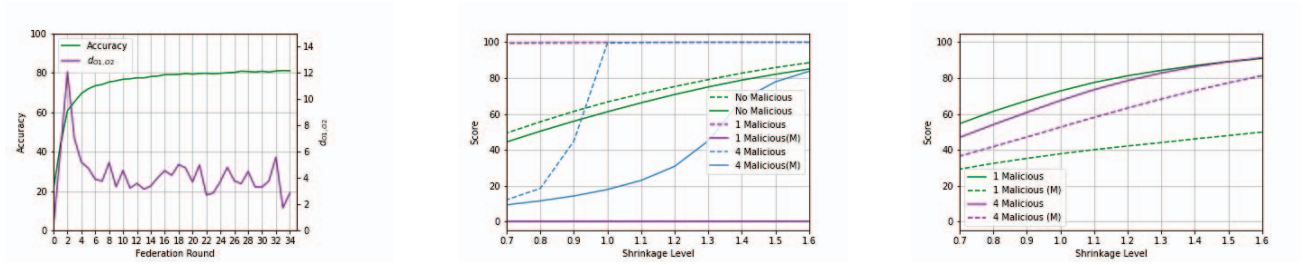


(a) $|M| = 1$ / $d_{O1,O2}$ is 50.84 for 4 LB, 43.22 for 8 LB, 37.40 for 12 LB

(b) $|M| = 3$ / $d_{O1,O2}$ is 36.54 for 4 LB, 29.47 for 8 LB, 24.14 for 12 LB

(c) $|M| = 4$ / $d_{O1,O2}$ is 18.38 for 4 LB, 11.69 for 8 LB, 8.69 for 12 LB

Fig. 7: Comparison of Clusters' Centroids / $|C| = 10$ / Label flipping attack with the increasing LB = 4, 8, 12.



(a) Cluster Difference $d_{O1,O2}$ and Accuracy during Federated Learning / $|C| = 10, |M| = 0$

(b) $|C| = 10$ / 100% poison rate / GF Attack / $d_{O1,O2}$: is 4.59 for $|M| = 0$, 99.69 for $|M| = 1$, 39.38 for $|M| = 4$

(c) $|C| = 10$ / Label Flipping Attack / $d_{O1,O2}$: is 28.38 for $|M| = 1$, 13.15 for $|M| = 4$

Fig. 8: Comparison of Clusters' Centroids / Fashion-MNIST

Figure 6a. When the number of malicious nodes increases in the federated learning system, they show a performance quite similar to regular ones (Figure 6b 6c). The same system has been tested with various numbers of quantized clients. Their performance is better than the ten quantized models. Hence, quantized and regular models can be combined in a single system and still malicious nodes can be detected.

For data poisoning experiments, the system is trained around 75% accuracy. Other training parameters are the same as model poisoning experiments except different local batch (LB) sizes as shown in Figure 7. The detection system clusters every client correctly, even at 40% maliciousness rate. The score difference between clusters decreases if the malicious rate or local batch size increases. The effect of the malicious device ratio is obvious since they modify the mean μ and standard deviation σ with their population. On the other hand, increased local batch size is an unexpected aspect of the detection system since the gap between clusters decreases with the increase of malicious devices. We may explain this aspect as client models become different with larger local batch sizes since each client's dataset may not be similar to other devices' datasets.

In Figure 8, similar tests are made with the Fashion-MNIST[25] dataset to compare the results. In each round, each client training is done with LB=3 in two epochs. Thus, the accuracy climbs up faster; however, the increase makes the system more vulnerable to data poisoning attacks since each model trains more between rounds. On the other hand, the

results are quite similar to the tests with the MNIST dataset.

V. Conclusion and Discussion

Multidimensional data points used in clustering and the pair-based comparison of their centroids developed in this work have proved efficient in various experiments. The framework that has been developed for experimental purposes can be used in real life with small modifications. The first advantage of the proposed method is to diverge every label and score them individually. The auto-encoder or total cosine similarity solution may not be able to catch the slight difference in small label attacks since every client may not have the iid data-set and same compute power. On the other hand, the follow-up studies may need to focus on various machine learning quantization techniques such as pruning or quantization-aware training with the real-world methods used in the industry. Moreover, other post-training quantization techniques may become more applicable in the future; thus, their impact could be investigated. Furthermore, the difference between slow/problematic learners and data poisoning attacks should be studied deeper so that the divergent ones could be more clear. On the other hand, we can assume that 13-15% of threshold γ is valid for all cases. However, if we increase the local batch size, this threshold may not meet the expectations. Then the label flipping attack may poison the global model. If the malicious client shows up in the system, it polarizes the malicious and benign clusters even with a low poisoning rate.

References

[1] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," International Conference on Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.

[2] J. Liu, J. Huang, Y. Zhou et al., "From distributed machine learning to federated learning: A survey," Knowledge and Information Systems, 2022, pp. 1–33.

[3] V. Rey et al., "Federated learning for malware detection in iot devices," Computer Networks, p. 108693, 2022.

[4] Y. Meidan et al., "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," IEEE Pervasive Computing, vol. 17, no. 3, pp. 12–22, 2018.

[5] T. D. Nguyen et al., "Flguard: Secure and private federated learning," 2021, arXiv:2101.02281.

[6] Y. Wang et al., "Rflbat: A robust federated learning algorithm against backdoor attack," 2022, arXiv:2201.03772.

[7] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," 2019, arXiv:1909.05125.

[8] T. Zhang et al., "Federated learning for internet of things," ACM Conference on Embedded Networked Sensor Systems, 2021, pp. 413–419.

[9] P. M. S. Sanchez et al., "Studying the robustness of anti-adversarial federated learning models detecting cyberattacks in iot spectrum sensors," 2022 arXiv:2202.00137.

[10] T. D. Nguyen et al., "Dïot: A federated self- learning anomaly detection system for iot," International Conference On Distributed Computing Systems (ICDCS), 2019, pp. 756–767.

[11] S. Li et al., "Abnormal client behavior detection in federated learning," 2019, arXiv:1910.09933.

[12] Y. Zhao et al., "Pdgan: A novel poisoning defense method in federated learning using generative adversarial network," International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), 2019, pp. 595–609.

[13] D. Li et al., "Detection and mitigation of label-flipping attacks in federated learning systems with kpca and k-means," International Conference on Dependable Systems and Their Applications (DSA), 2021, pp. 551–559.

[14] T. D. Nguyen et al., "Poisoning attacks on federated learning-based iot intrusion detection system," Workshop on Decentralized IoT Systems and Security (DISS), 2020, pp. 1–7.

[15] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," Annual Conference on Computer Security Applications (ACSAC), 2016, pp. 508–519.

[16] A. Reisizadeh et al., "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2021– 2031.

[17] F. Haddadpour et al., "Federated learning with compression: Unified analysis and sharp guarantees," International Conference on Artificial Intelligence and Statistics, PMLR, 2021, pp. 2350–2358.

[18] Y. Liu et al., "Communication-efficient federated learning for anomaly detection in industrial internet of things," Global Communications Conference (GLOBECOM), 2020, pp. 1–6.

[19] Post-training quantization : Tensorflow Lite. https://www.tensorflow.org/lite/performance/post_training_quantization

[20] F. Chollet et al., Keras, https://github.com/fchollet/keras

[21] TensorFlow Datasets, a collection of ready-to-use datasets, https://www.tensorflow.org/datasets

[22] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1–9.

[23] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," 2019, arXiv:1907.02189.

[24] Y. LeCun, C. Cortes, and C. Burges, "Mnist hand-written digit database," ATT Labs, vol. 2, http://yann.lecun.com/exdb/mnist

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