Detection and Mitigation of Label-Flipping Attacks in Federated Learning Systems with KPCA and K-Means

Dongcheng Li
Department of Computer Science
University of Texas at Dallas
Richardson, USA
dxl170030@utdallas.edu

W. Eric Wong*
Department of Computer Science
University of Texas at Dallas
Richardson, USA
ewong@utdallas.edu

Wei Wang
China University of Geosciences
Wuhan, China
wangwei5811@foxmail.com

Yao Yao
China University of Geosciences
Wuhan, China
905345546@qq.com

Matthew Chau
University of Texas at Dallas
Richardson, USA
mcc180003@utdallas.edu

Abstract—**Federated learning is a popular machine-learning technique that is often preferred due to its efficiency and data privacy. However, federated-learning systems face a serious threat of data poisoning that can cause the systems and predictions to fail if not treated in time. This study extends another study of data-poisoning attacks in federated-learning systems conducted by Tolpegin et al. We first investigate the effectiveness of the defense strategy suggested in Tolpegin's study. Then we propose an improved defense strategy that emphasizes employing KPCA and K-mean clustering. It is demonstrated in this paper that our defense strategy, when combined with improved dimensionality-reduction algorithms, produces better results in defending against data-poisoning attacks in federated-learning systems.**

Keywords: federated learning; data poisoning; label flipping; KPCA

I. INTRODUCTION

Federated learning (FL) is a popular machine-learning (ML) technique that is often preferred over traditional ML because it is faster, more efficient, and offers greater data privacy. However, FL systems face a serious threat of data poisoning that can cause the systems and predictions to fail if they are not treated early on [1].

This paper aims to contribute a defense strategy for FL systems because current FL systems are very vulnerable to data-poisoning attacks. In this paper, we examine the effectiveness of Tolpegin's defense strategy against data-poisoning attacks in FL systems. Then we propose an improved defense strategy which uses kernel principal component analysis (KPCA) and K-mean. Lastly, we compare the results of the experiments on the effectiveness of different defense strategies in a controlled environment.

The organization of this paper is as follows. In Section 2, the existing literature is reviewed which relates to the detection and mitigation of data poisoning in FL systems. A high-level overview of the FL system, data-poisoning attacks, and clustering techniques is presented in Section 3. In Section 4, we described the setup for the experiment along with the proposed defense solution. In Section 5, findings from the experiment are explained on the basis of the resulting outputs. Finally, we draw the conclusion and describe future work that can be done on the area.

II. RELATED STUDIES

The trustworthiness of the new model is a considerable problem, as ML is being applied to every field. Therefore, trustworthiness needs to be considered before we can believe in the results of ML [1]. Traditional ML is slower than FL because the new model can be trained faster and more efficiently and the privacy of the raw data is ensured. However, few relevant studies have been done on counter-poisoning attacks on FL; instead, the focus is on traditional ML [1]. With increase in the popularity of FL, there is a need to work on its vulnerabilities, as there are various FL attacks: e.g., backdoor attacks, gradient-leakage attacks, and membership-interference attacks [2]-[10].

There are two types of poisoning attacks in FL: model poisoning and data poisoning. This study falls in the data-poisoning category. An aggressor can manipulate training-data labels by using the label-flipping attack, thereby poisoning the data. The model is not interfered with anyway, and it runs smoothly [1]. On the other hand, model-poisoning attacks target the model, leading to high error rates in the model and causing the FL system to fail. However, model poisoning is hard to implement, as it requires expert poisoning participants to execute successfully. Data poisoning, on the other hand, can be done by non-expert participants [1]. Therefore, data poisoning is preferable and is used more frequently, as it can be done easily.

To detect and mitigate various FL attacks, Chen et al. [11] introduced a federated-pruning method to remove redundant neurons in the network and adjust the model's extreme weight values. Fung et al. [12] proposed a novel defense for FL Sybil attacks that identifies poisoning Sybils based on the diversity of client updates in the distributed-learning process. Prakash and Avestimehr [13] proposed a strategy to mitigating FL Byzantine behaviors in heterogeneous data-distribution settings by comparing each client's update with a guiding update of that client. Fu et al. [14] designed an aggregation algorithm which combines repeated median regression with a reweighting scheme in iteratively reweighted least squares. In addition, Tahmasebian et al. [15] proposed a robust aggregation algorithm inspired by truth-inference methods used in crowdsourcing by incorporating the worker's reliability into aggregation. Moreover, some studies have considered data-poisoning attacks and their defense systems that are more closely related to ML systems

2767-6684/21/$31.00 ©2021 IEEE
DOI 10.1109/DSA52907.2021.00081

and less concerned with the FL system that is under consideration [16]. These studies examined defense systems like server-side defense and anomaly detection/k-NN. Other related works have analyzed different types of data-poisoning attacks such as spam filtering [17], malware and network anomaly detection [18], [19], disease diagnosis [20], computer vision [21], and recommender systems [22].

III. Preliminaries

A. *Federated Learning*

Machine learning has become a popular tool for predicting outcomes, whereas federated learning is an ML technique. Since the prediction of ML depends upon the data provided to it, and since large amounts of data increase, the prediction-accuracy management of data becomes a cumbersome task [1]. Another problem associated with ML is how can the data models be considered accurate, that is, how trusted they are. Privacy of the data also causes concern when data is collected for the model. Federated learning offers a solution to these problems because, instead of sharing raw data, participants need to share only their model parameters [1]. Therefore, a global data model is created wherein all the individual participants send their model parameters to a central server and are made through this collection. This makes FL much more efficient and faster than traditional ML, since each participant only needs to train his or her local dataset. The global model is updated by aggregating the local dataset model updates sent by the participants [1]. Since the aggregation of the individual participants makes this dataset, there is no centralized server curator to verify it. As such, the FL system is left vulnerable to poisoning attacks.

B. *Vulnerability in FL System*

The scenario considered in the experiment involves a group of participants in the FL system who under the influence of a malicious adversary. The percentage of the participants under control of the malicious adversary can be denoted as *m*. Those participants are used to poison the global model for a set number of rounds in the FL system. Rather than target all the model's classes, the adversary's objective is to alter the learned parameters for a specific class so that the final model, *M*, has a high error rate [1]. As a result, the adversary's attack is a targeted poisoning attack instead of an untargeted attack, which aims for high errors across all classes in the global model [1], [23]-[25]. Targeted attacks have the advantage of being hard to detect, as their influence is limited only to the class they affect and all other classes function normally.

C. *Label-Flipping Attacks*

In label-flipping, also known as data poisoning, an attacker can control the labels assigned to the training data. By flipping the labels, the performance of the FL system can be seriously diminished even if only a fraction of the training-point labels have been flipped. For example, changing the labels of the *airplane* class to that of the *dog* class in the CIFAR-10 database is a successful label-flipping attack. Since an FL system can have *m* of malicious participants, this label-flipping attack can cause the global model to have a high error rate.

There is no centralized authority to validate data in the FL, which makes it vulnerable to data poisoning. Data poisoning can be done by aggressors who appear in the form of participants on the FL system. These may either have malicious intentions or may have been compromised by some adversary [1]. The training-data updates provided by them can be mislabeled or may have poisonous samples. In the absence of a central authority, the updates provided by adversaries do not get filtered and, as a result, poison the global model being trained. This data poisoning is undetectable through ordinary means but can be found through dimension-reduction and clustering algorithms [1]. Using these, the updates of honest participants can be differentiated from those of aggressors by saving the data from being poisoned.

IV. Analysis of Defense Strategies Against Label-Flipping Attacks in FL

This study extends Tolpegin et al.'s study of data-poisoning attacks of FL systems. From Tolpegin's study, we see the effectiveness of a targeted poisoning attack against the FL system and the potential for a dimensionality algorithm to defend against such poisoning attack. We improve upon Tolpegin's study by using a specific dimensionality-reduction algorithm with clustering and proving its superiority. More specifically, we use KPCA instead of principal component analysis (PCA) because of KPCA's advantages over the latter, and we use k-mean clustering for noise reduction. In addition, to render the experiment controlled and accurate, the same framework and dataset used by Tolpegin et al. were also adapted for our study.

A. *Federated-Learning System Setup*

In the experiment, the FL system is implemented in the PyTorch library available in the programming language Python [25]. There is one central aggregator and N-50 participants by default in the setup, k = 5. The distribution of the training dataset among the participants is uniform and random relative to the total training dataset. Furthermore, each participant is assumed to have a distinct set of training data. We applied the independent distribution methodology and used identically distributed data (IID) on the data to accomplish this. Each participant *Pi*'s training dataset *Di* contains no testing data, as testing data is used to validate the model. Hence, it is not required to be given to the participants. The FL experiment is set to run for R=200 rounds in total because it has been observed that the DNN coverage of both models is less than 200 training rounds.

B. *DNN Architecture Setup*

The CIFAR-10 [26] and Fashion-MNIST [27], two famous image-classification datasets, have been used in the current scenario. There are 10 object classes in CIFAR-10, each of which has 6,000 images for a total of 60,000 images, all of which are all colored [1]. The CIFAR1-10 has object classes such as frogs, horses, and airplanes [26]. The CIFAR-

10 dataset is divided into batches of 10,000 images, with five batches kept for training data and one kept for training the model. The same is seen in Fashion-MNIST, which is divided into six batches, each with 10,000 images. Similarly, five are kept for training the dataset, and the remaining one is used for validating the model. The images in Fashion-MNIST are greyscale and are associated with 10 classes of clothing such as dresses, shirts, and sneakers [27]. Tables 1 and 2 show the DNN architecture used in both the datasets:

TABLE I. THE DNN ARCHITECTURE USED IN CIFAR1-10

Layer Type	Size
Conv+Relu+Batch Norm	3*3*32
Conv+Relu+Batch Norm	3*32*32
Max Pooling	2*2
Conv+Relu+Batch Norm	3*3*32
Conv+Relu+Batch Norm	3*32*32
Max Pooling	2*2
Conv+Relu+Batch Norm	3*3*32
Conv+Relu+Batch Norm	3*32*32
Max Pooling	2*2
Fully Connected	2048
Fully Connected+Softmax	128/10

TABLE II. THE DNN ARCHITECTURE USED IN FASHION-MNIST

Layer Type	Size
Conv+Relu+Batch Norm	5*1*16
Max Pooling	2*2
Conv+Relu+Batch Norm	5*16*32
Max Pooling	2*2
Fully Connected	1568/10

The DNN architecture used for CIFAR-10 has a test accuracy of ~78% without poisoning. From the table, we can see that, to accomplish this in CIFAR-10, we use six convolutional layers, batch normalization, a rectified linear activation function (ReLU), three max-pooling layers, and two fully connected dense layers with one running SoftMax. For the Fashion-MNIST setup, we use a two-layer convolution network, batch normalization, ReLU, two max-pooling layers, and one fully connected dense layer. The resulting DNN architecture, in this case, has a test accuracy of ~91% without poisoning.

C. *Label-Flipping Attack Setup*

For the experiment to mimic a label flipping attack in an FL system with *N* participants, of which *m* are malicious. We first randomly assigned a number of participants (*N*) for the experiment. Then *N x m* of participants (*P*) were randomly identified as malicious at the start of each experiment, and the remaining were identified as honest. However, to make the experiment more accurate, we considered the effect caused by malicious participants chosen randomly; so, each of the experiments were replicated 10 times, and the average result was taken as the final value. As a result, the value of *m* is set to 10 percent in the experiment (that is, $m = 10\%$). Three label-flipping attacks settings were explored in the experiment to represent a broad set of conditions an adversary could use to approach an FL system to attack. The following were the conditions used in the experiment:

- In the first case, the source class was very frequently misclassified as the target class.
- In the second case, the source class was very infrequently misclassified as the target class.
- The third case is a combination of the first and second cases.

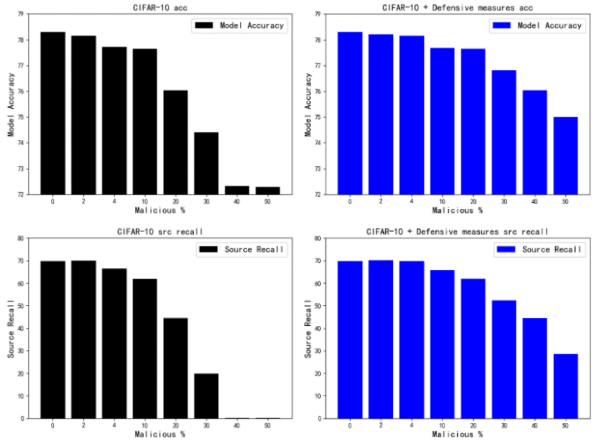
The class labels used from CIFAR-10 for testing for the first case were dog to cat. In the second case, we used airplane to bird labels; and in the third case, we tested the automobile to the truck. In the case of Fashion-MNIST, the class labels used for the first one were shirt to t-shirt/top. Trouser to dress were used for the second case and coat to the shirt were used for the third case.

D. *Defense Strategy Feasibility*

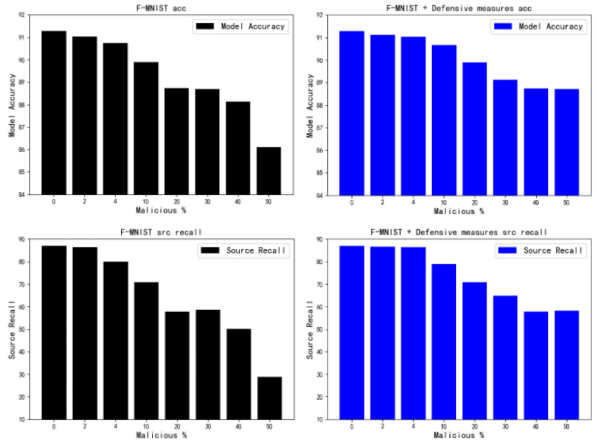
We first checked the defense strategy employed by Tolpegin et al. and then introduced our proposed defense strategy. A defense strategy needs to be implemented so that the FL system can defend against the label-flipping attack discussed so far. The defense needs to be such that it can defend against highly effective adversaries. For this, an algorithm needs to be introduced in the FL system so that the aggregator can identify malicious participants. By using the algorithm proposed in Tolpegin's study [1], an aggregator can identify the malicious participants in the FL system. After these malicious participants have been identified, the aggregator can either choose to blacklist them or ignore their updates for the upcoming rounds.

The basis of the defense strategy used is that a dimensionality-reduction algorithm can catch malicious updates because the parameter updates contain unique characteristics. However, since DNNs have many parameters, manually checking for these malicious parameter updates is challenging. In contrast, as seen in the paper, an automated approach can use PCA, which is a dimensionality-reduction algorithm, to find and filter the parameters sent by the malicious updates.

Figure 1 shows the results generated from the experiment regarding the impact on the number of malicious participants in the global model.



a) CIFAR1-10



b) Fashion-MNIST

Figure 1. The source recall and global model accuracy obtained by attacking CIFAR-10 and Fashion-MNIST by the *m*% adversaries

The factors considered here are the percentage of malicious participants, that is, the *m* of the participant effect on the global model accuracy and source class recall. The *m* used for this step ranges from two to 50 percent. From the results, it can be seen that as *m* increases, the test accuracy of the global model decreases. Furthermore, even with a small *m*, the global-model test accuracy still decreases compared to that of a non-poisoned model, but the source-class recall of the model exhibits an even more significant decrease in this case. When *m* is 40%, the global model test accuracy drops from 78.3% in the case of the non-poisoned model to 74.4% in the poisoned scenario for CIFAR-10. Similarly, the source-class recall falls to 0%. Fashion-MNIST shows a similar case of global-model test accuracy and source-class recall. For example, when *m* is four percent, the source class decreases by about 10 percent. By this, we can see that even if an adversary controls a very small percentage of participants, he or she can still cause the global model accuracy to drop. Thus, even a few participants under the control of an adversary can significantly impact an FL system. Though both datasets are vulnerable to label-poisoning attacks, there is a difference in the degree of vulnerability. From the results of the experiment, it can be seen that CIFAR-10 is more vulnerable than Fashion-MNIST. During the experiment, we also found that it is not essential for the adversary to identify the most vulnerable source and target-class combination. Because there is not necessarily a correlation between attack effectiveness and misclassification performance for the non-poisoned model, we can see from the above section that, after the elimination of malicious participation, a high-utility convergence can eventually be achieved. The possibility of such a recovery from early-round attacks supports the case for using the proposed detection technique as a defensive strategy. This finding is consistent with Tolpegin's study. The result shown above also verifies that Tolpegin's proposed algorithm indeed reduces the effect of label-flipping attacks on FL systems.

E. Defending Against Label-Flipping Attacks with KPCA

What follows is the algorithm to be used instead of the one proposed by Tolpegin's study [1]:

Algorithm 1: KPCA

```
function kpack (x, sigma, cls, target-dim)
  #Data extraction
  psize ← size(x)
  m ← psize(1)
  n ← psize(2)
  #Sampleing kernel matrix
  l ← ones(m, m)
  for i ∈ (1,m) do
        for j ∈ (1,m) do
           k(i,j) ← kernel (x(i,:),x(j,:),cls, sigma)
  #Computing kernel absolute value after extraction
  k1 ← k-1*k/m-k*l/m+l*k*l/(m*m)
  #calculate signal values and eigenvalues
  [v, e] < eig(k1)
  e ← diag (e)
  # Filtering eigenvalues and eigenvectors
  [dump, index] ← sort(e, 'descend')
  e ← e(index)
  v ← v(:,index )
  rank ← 0
  for i ∈ (1,size(v, 2)) do
        if e(i) < le − 6  then
              break
        else
              v(:, i) ← v(:,1) . /sqrt(e(i))
        rank ← rank + 1
  eigenvectors ← v(:,1: target_dim )
  eigenvalues ← e(1: target_dim )
  #projection
  project_invectors ← k1* eigenvectors
```

In Algorithm 1, we first performed data extraction on KPCA to obtain the number of samples m, the sample dimension n, and a matrix with m rows, and n columns with weighting value of 1. Use it to calculate the kernel matrix *K* according to the kernel formula, then centralize *K* to obtain the centralized kernel matrix *K*l, and then calculate the eigenvalue *v* and eigenvector *e* of the decentralized *K*l. Next sort *e* in descending order, then filter the eigenvalues and eigenvectors by traversing each column of eigenvalue *v*. Finally, calculate the project intectors of the centered kernel matrix *k*1 on the eigenvectors.

Algorithm 2 enables the aggregator to identify malicious participants. Let *R* denote the vulnerable FL training round set. Let *P* denote the participant set and *PT* denote the current global model parameters. For each round $r \in R$, find the set of participants *Pr* queried in the participant set *P* of this round and the global model parameter *PTr*-1 of the previous round. Then update the model parameters for each participant after training the DNN, and at the same time the aggregator calculates the incremental $PT_{\Delta,i}$ by comparing the

participant-model update with the global-model update. Then connect the parameters in $PT_{\Delta,i}$ to the source-class output node and add the result to *PT*. Finally, obtain the result via standardization, dimensionality reduction, and clustering on *PT*.

Algorithm 2: An improve algorithm for identify malicious model updates in FL using KPCA

def evaluation update (R: set of vuluerable train rounds, P: participant set, U: update)

$PT =$ current global model parameter

for each round $r \in R$ do

P_r: participants in P queried in train round r

PT_{r-1} : Global model parameters after training round $r-1$

for participant $P_i \in P_r$ do

$PT_{r,i}$: updated parameters after train DNN (PT_{r-1})

$PT_{\Delta,i} = PT_{r,i} - PT_r$: "Aggregator computes delta in participant's model update compared to the global"

$PT\frac{src}{\Delta,i}$: parameter in $PT_{\Delta,i}$ connect to source class output node

Add $PT\frac{src}{\Delta,t}$ to PT

$PT' =$ standardize (PT)

$PT'' =$ dimensionality_reduction (PT')

$PT''' = cluster(PT'')$

$Plot(PT'', PT''')$

Using this algorithm, we can successfully identify the malicious participants in the FL system. In the case of PCA, this algorithm is used for dimensionality reduction when the feature space contains too many irrelevant or redundant features. The aim is to find the intrinsic dimensionality of the data [28]. The KPCA, on the other hand, can be considered a smaller version of PCA [28], [29], but KPCA has advantages over PCA. The PCA performs better on linear data. As such, it can handle skewed data very effectively when its performance with non-linear data is not up to the mark [30]. While this may not be a problem in the beginning, it becomes difficult for PCA to identify malicious updates if the dataset has non-linear data. On the other hand, KPCA can handle non-linear data very effectively while also successfully handling linear data (such as skewed data) [30], [31]. This was seen when both PCA and KPCA were applied to the ECG200 dataset for a classification comparison. The KPCA separated the two classes best [30]. The comparison was a made on a number of factors. For example, the quality of data separated, the dimension, the F1 score, the runtime, a power analysis and the *p* value were all considered. Another factor that gives KPCA an edge over PCA is the topology used for random projection, which can sometimes cause the projection to be less than optional in the scenario [30]. In the case of KPCA, a manifold topology is used, which allows it to detect data discrepancies in the lower dimension [30]-[32]. This makes KPCA easy to implement and a better choice than PCA for the detection of malicious updates, as it covers the weakness of the PCA (i.e., PCA can fail to detect malicious updates in non-linear data).

K-means is a clustering algorithm that returns a natural grouping of data points based on their similarity. As such, it can be considered a special case of Gaussian mixture models [28]. While both K-mean clustering and PCA first seem to have very different objectives, in reality, both of them exhibit a very deep connection [29]. The connection is such that K-mean can be considered a scattered PCA. The PCA aims to minimize the mean-squared reconstruction error by representing all data vectors as a linear combination of a small number of eigenvectors [29]. The K-mean also aims to minimize the mean-squared reconstruction error by representing all data vectors as linear combinations of a small number of cluster centroids [29]. It is a common practice to apply PCA before a clustering algorithm such as K-mean. It is believed that doing so improves the clustering results (that is, reduces noise) in practice.

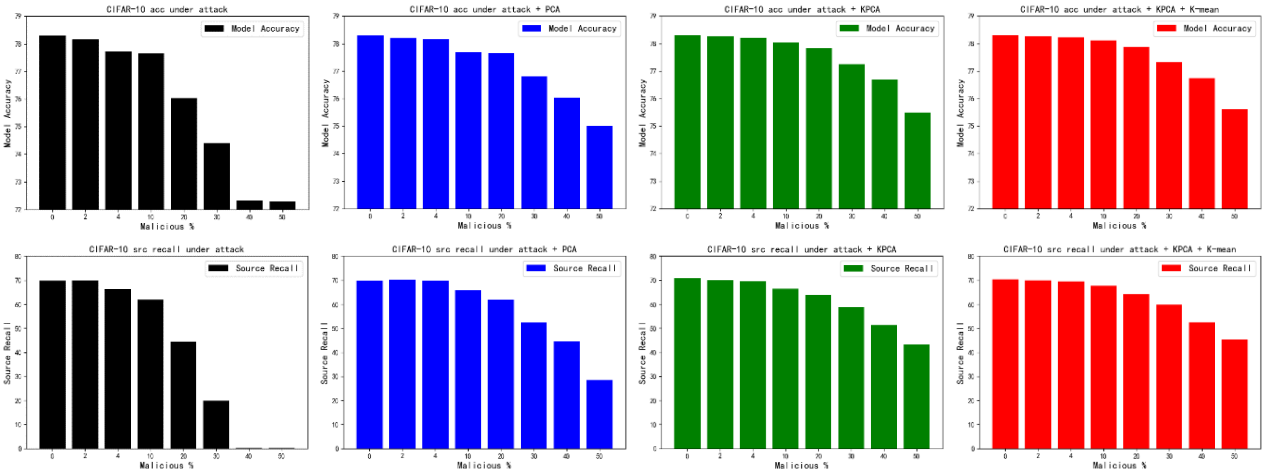
F. Experiments Setup

In the experiment, the CIFAR-10 and Fashion-MNIST were attacked by an adversary using the methods discussed above. The results are shown in a bar graph for four cases. For the first case, no dimensionality-reduction algorithm was applied the dataset. For the second, only PCA was applied as the defense algorithm. The third case used KPCA as a defense method, and the fourth used KPCA along with K-mean as the method for defense against adversarial attacks on the database. The same settings were applied to check both accuracy and global models, so each has four related graphs. The graphs are also represented by four colors, and each of the cases has one. Black is for Case 1, blue is for Case 2, green is used to represent Case 3, and Case 4 is represented by red.

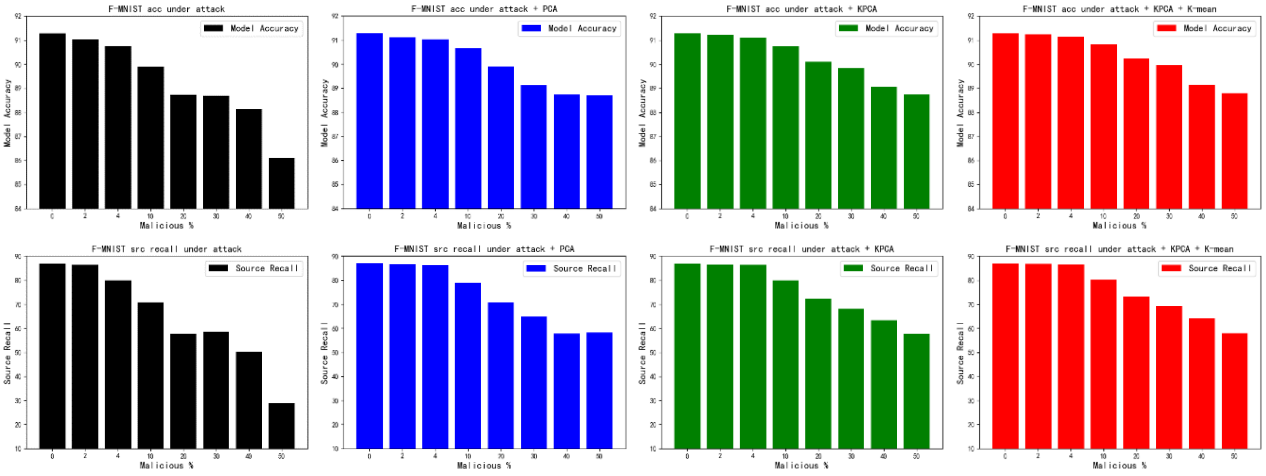
On the other hand, the second experiment was conducted to check how well the proposed algorithm (Algorithm 2). The two-dimensional plot shows how effective the algorithm can be when using KPCA and K-means to help identify malicious updates sent by adversaries from honest updates. The blue color in the plot is for the malicious updates, and yellow is used to denote the updates sent by honest participants. The plots are drawn from the output generated by the proposed algorithm.

V. RESULTS COMPARISON AND ANALYSIS

Figure 2 shows bar graphs for two datasets for the data-poisoning effect on source recall and model accuracy. The values for *m* (that is, the number of malicious participants for both experiments) were set from zero to 50 percent. For model accuracy, the values were between 72 to 79 in CIFAR-10, while the source-recall values for CIFAR-10 are between zero to 80. In Fashion-MNIST, the values for model accuracy were between 84 and 92. Source recall for Fashion-MNITS has values between 10 and 90. The graphs offer a comparison of four states (that is, when the FL system is not provided with any defense from data poisoning). The results for both of the datasets are discussed below.



a) CIFAR1-10

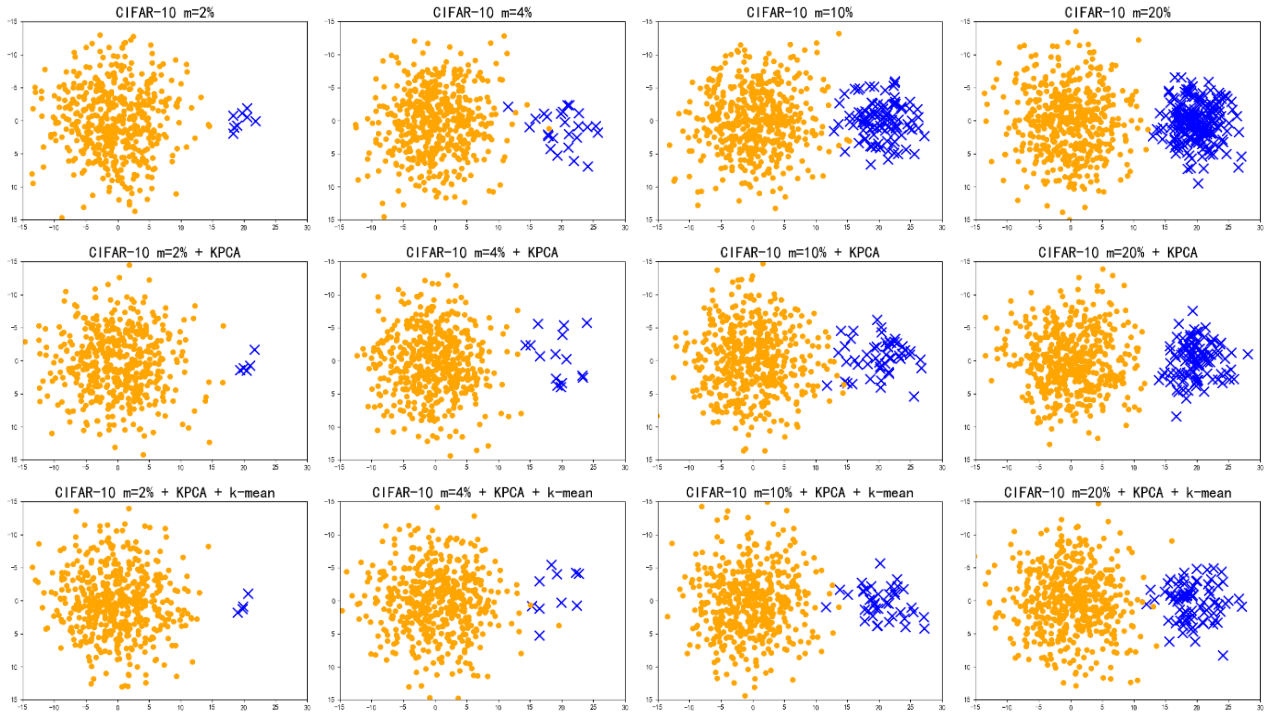


b) Fashion-MNIST

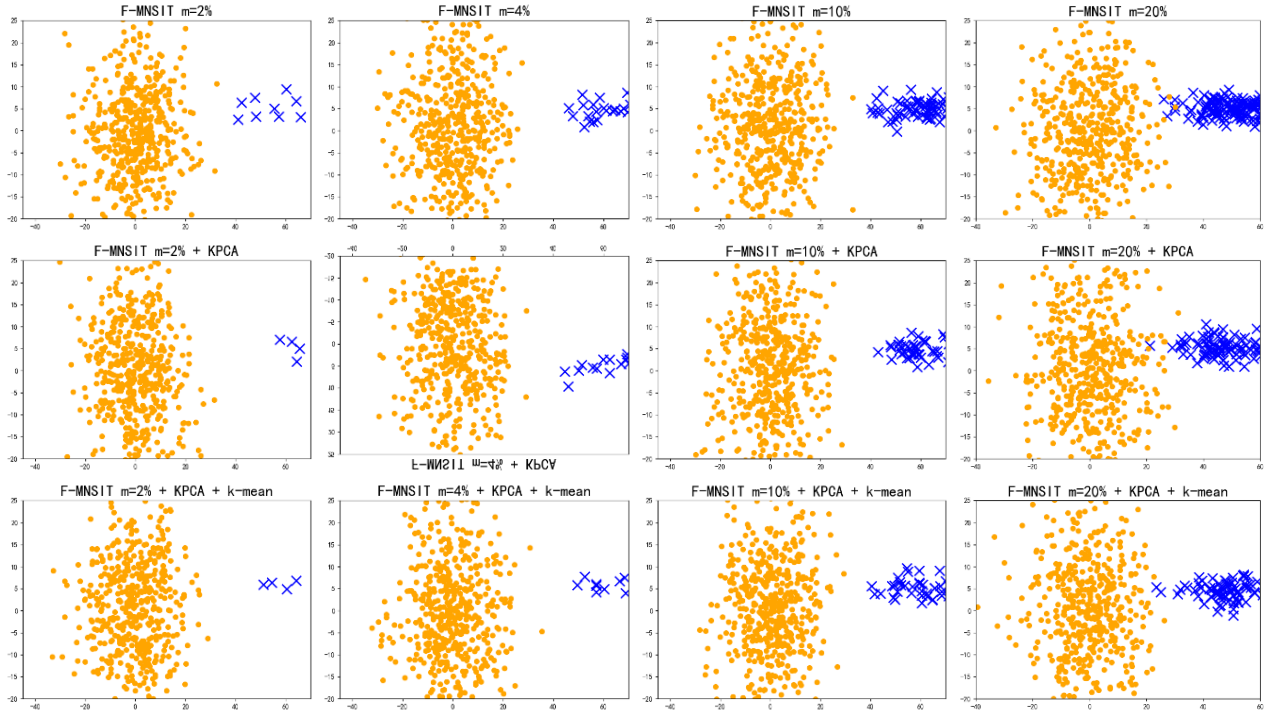
Figure 2. The source recall and global model accuracy obtained by attacking the CIFAR-10 and Fashion-MNIST by *m* adversaries under various defend strategies

In the first case, it can be seen from the graph in black color that, with an increase in the number of malicious participants, the accuracy falls from the original 78 percent. The fall is at first gradual, but when the percentage of malicious participants increases from 10 percent, the accuracy drops sharply. From 40 to 50 percent, the accuracy does not drop any further and stays at a constant low point. The source recall of the model suffers the same fate when the percentage of malicious participants in the FL system is greater than 10 percent, and it drops sharply. Moreover, after the percentage of malicious participants is greater than 30 percent, the source recall of the model falls to zero. The second case is depicted in the blue graph on the occasion when PCA is used in the defense against data poisoning. From the graph, it can be seen that there is a minimal fall in accuracy when the percentage of malicious participants is two, and there is no further drop in accuracy until the percentage is increased to four. After four percent, there is a sharp drop to 10 percent, and then there is no further drop in accuracy until 20 percent. However, after 20 percent, there are sharp falls for every 10 percent increase in the number of malicious participants. The third case is for using KPCA instead of PCA and is represented by the green graph. The green graph shows that, compared to the blue one (that is, when PCA was used), there is a smaller drop in accuracy compared. When KPCA is applied, there is no drop in

accuracy until the percentage of malicious participants is greater than four percent, while in PCA, the drop in accuracy starts from when the percentage is greater than two percent. The red graph (that is, KPCA with K-means) shows a similar result, with a slight increase in reduction of accuracy. However, K-means is a time-consuming process; hence, it is not the optimal choice. A similar trend can be seen in the source recall. As such, it can be seen that KPCA performs best for defending against data-poisoning attacks.



a) CIFAR1-10



b) Fashion-MNIST

Figure 3. The effectiveness of the proposed algorithm in distinguishing malicious updates from honest updates

The same scenario as the one discussed in CIFAR-10 can be seen as compared to the other graphs the green one that is KPCA performs best as the drop in accuracy and recall is the lowest, thereby giving further support to the theory that the proposed defense strategy counters the weakness of the FL system with respect to data-poisoning attacks. We can see from the green and red graphs that the use of the K-mean cluster algorithm after the KPCA brings no significant improvement for clustering results. Due to the added complexity and time for computing of the K-mean cluster, our defense strategy may not benefit with the K-mean cluster added.

Figure 3 shows the ability of the proposed defense algorithm and depicts how it compares with that proposed by Tolpegin [1].

In the first row, we can see the results when the PCA defense algorithm is used. In the second row, we can see the algorithm with KPCA; and in the third row, we have the algorithm used in KPCA but with K-means added. From the plots figure, it can be seen that the malicious updates differ from honest updates and how much the different algorithms can differentiate between them. Comparing the three rows, it can be seen that KPCA differentiates better than PCA and that there is little difference between using K-mean and not using it. Thus, using KPCA in the defense algorithm is the better course of action.

VI. Conclusion and future direction

This paper reports on a successful attempt to detect and mitigate data-poisoning attacks in FL by employing dimensionality-reduction and clustering techniques. FL is a crucial system which is preferred due to the distributed nature of its networks. However, the distributed property of FL poses serious threats to data poisoning, as data in an FL system is not shared across the central server while malicious participants and malign activities can be overlooked. For this purpose, the research carried out by Tolpegin and others has provided a guiding path to the resolution of data poisoning within the FL system. PCA can help the FL system to identify malicious attempts. However, this solution is not sufficient to rebut the threats posed by data-poisoning attacks. For this purpose, this study projects the use of KCPA and K-Means instead of PCA. The result of the experiment shows that using KPCA instead of PCA is more effective in defense against poisoning attacks in an FL system. In the future, we will conduct experimental research on other defense strategies concerning the remaining types of data-poisoning attacks, such as the backdoor attack. Also, we will explore and analyze the empirical basis of the effectiveness of other dimensionality-reduction algorithms for mitigating vulnerabilities in the FL system.

Reference

[1] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," *Computer Security – ESORICS 2020*, pp. 480–501, 2020.

[2]C. Xie, K. Huang, P. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," In *International Conference on Learning Representations*. 2019.

[3] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," In *S. Chiappa & R. Calandra (Eds.), Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, vol. 108, pp. 2938–2948, 2020.

[4] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" ArXiv, 2019.

[5] W. Wei, L. Liu, M. Loper, K.H. Chow, M.E. Gursoy, S. Truex, and Y. Wu, "A framework for evaluating gradient leakage attacks in federated learning," arXiv, 2020.

[6] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan," In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[7] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," 2019 IEEE Symposium on Security and Privacy (SP), 2019.

[8] L. Zhu and S. Han, "Deep leakage from gradients," Lecture Notes in Computer Science, pp. 17–31, 2020.

[9] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," 2019 IEEE Symposium on Security and Privacy (SP), pp. 739–753, 2019.

[10] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Demystifying membership inference attacks in machine learning as a service," IEEE Transactions on Services Computing, 2019.

[11] C. Wu, X. Yang, S. Zhu, and P. Mitra, "Mitigating backdoor attacks in federated learning," arXiv, 2020.

[12] C. Fung, C.J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," arXiv, 2018.

[13] S. Prakash and A.S. Avestimehr, "Mitigating byzantine attacks in federated learning," arXiv preprint arXiv:2010.07541, 2020.

[14] S. Fu, C. Xie, B. Li, and Q. Chen, "Attack-resistant federated learning with residual-based reweighting," arXiv preprint arXiv:1912.11464, 2019.

[15] F. Tahmasebian, J. Lou, and L. Xiong, "RobustFed: a truth inference approach for robust federated learning," arXiv preprint arXiv:2107.08402, 2021.

[16] A. Paudice, L. Muñoz-González, and E. C. Lupu, "Label sanitization AGAINST LABEL Flipping poisoning attacks," ECML PKDD 2018 Workshops, pp. 5–15, 2019.

[17] B. Nelson, M. Barreno, F.J. Chi, A.D. Joseph, B.I. Rubinstein, U. Saini, C. Sutton, J.D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," LEET, 8, pp.1-9, 2008.

[18] S. Chen, M. Xue, L. Fan, S. Hao, L. Xu, H. Zhu, and B. Li, "Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach," Computers & Security, vol. 73, pp. 326–344, 2018.

[19] B.I. Rubinstein, B. Nelson, L. Huang, A.D. Joseph, S.H. Lau, S. Rao, N. Taft, and J.D. Tygar, "Antidote: understanding and defending against poisoning of anomaly detectors," In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, pp. 1-14, 2009.

[20] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "Systematic poisoning attacks on and defenses for machine learning in healthcare," IEEE Journal of Biomedical and Health Informatics, vol. 19, no. 6, pp. 1893–1905, 2015.

[21] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," 2018 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 399–414, 2018.

[22] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," Proceedings of the 34th

Annual Computer Security Applications Conference, pp. 381–392, 2018.
[23] B. Biggio, B. Nelson, and P. Laskov, “Poisoning attacks against support vector machines,” arXiv preprint arXiv:1206.6389, 2012.
[24] M. Fang, X. Cao, J. Jia, and N. Gong, “Local model poisoning attacks to byzantine-robust federated learning,” In 29th Security Symposium, pp. 1605-1622, 2020.
[25] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, “Is feature selection secure against training data poisoning?” In international conference on machine learning, pp. 1689-1698, 2015.
[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and A. Desmaison, “Pytorch: An imperative style, high-performance deep learning library,” Advances in neural information processing systems, 32, pp.8026-8037, 2019.
[26] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009.
[27] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” arXiv preprint arXiv:1708.07747, 2017.
[28] D. Napoleon and S. Pavalakodi, “A new method for dimensionality reduction Using KMEANS clustering algorithm for high dimensional data set,” International Journal of Computer Applications, vol. 13, no. 7, pp. 41–46, 2011.
[29] C. Ding and X. He, “K-means clustering via principal component analysis,” Twenty-first international conference on Machine learning - ICML '04, vol. 29, 2004.
[30] F. Anowar, S. Sadaoui, and B. Selim, “Conceptual and empirical comparison of dimensionality reduction Algorithms (PCA, Kpca, LDA, MDS, SVD, LLE, Isomap, LE, ICA, t-SNE),” Computer Science Review, vol. 40, p. 100378, 2021.
[31] L. Van Der Maaten, E. Postma, and J. Van den Herik, “Dimensionality reduction: a comparative,” J Mach Learn Res, 10(66-71), p.13, 2009.
[32] A. García-González, A. Huerta, S. Zlotnik, and P. Díez, “A kernel Principal Component Analysis (kPCA) digest with a new backward mapping (pre-image reconstruction) strategy,” 2020.