Confident federated learning to tackle label flipped data poisoning attacks

Pretom Roy Ovi, Aryya Gangopadhyay, Robert Erbacher, Carl Busart

Pretom Roy Ovi, Aryya Gangopadhyay, Robert F. Erbacher, Carl Busart, "Confident federated learning to tackle label flipped data poisoning attacks," Proc. SPIE 12538, Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications V, 125380Z (12 June 2023); doi: 10.1117/12.2663911

Event: SPIE Defense + Commercial Sensing, 2023, Orlando, Florida, United States

Confident Federated Learning to Tackle Label Flipped Data Poisoning Attacks

Pretom Roy Ovi[a], Aryya Gangopadhyay[a], Robert F. Erbacher[b], and Carl Busart[b]

[a]Center for Real-time Distributed Sensing and Autonomy
[a]University of Maryland, Baltimore County, USA
[b]U.S. DEVCOM Army Research Laboratory, USA

ABSTRACT

Federated Learning (FL) enables collaborative model building among a large number of participants without revealing the sensitive data to the central server. However, because of its distributed nature, FL has limited control over the local data and corresponding training process. Therefore, it is susceptible to data poisoning attacks where malicious workers use malicious training data to train the model. Furthermore, attackers on the worker side can easily manipulate local data by swapping the labels of training instances to initiate data poisoning attacks. And local workers under such attacks carry incorrect information to the server, poison the global model, and cause misclassifications. So, detecting and preventing poisonous training samples from local training is crucial in federated training. To address it, we propose a federated learning framework, namely Confident Federated Learning to prevent data poisoning attacks on local workers. Here, we first validate the label quality of training samples by characterizing and identifying label errors in the training data and then exclude the detected mislabeled samples from the local training. To this aim, we experiment with our proposed approach on MNIST, Fashion-MNIST, and CIFAR-10 dataset and experimental results validated the robustness of the proposed framework against the data poisoning attacks by successfully detecting the mislabeled samples with above 85% accuracy.

Keywords: Federated Learning, Data Poisoning Attacks, Adversarial Attacks, Label Flipping Attacks

1. INTRODUCTION

Distributed machine learning has been a topic of interest for many years. Distributed machine learning offers the advantage of processing large amounts of data by distributing the workload among multiple machines.[1,2] It can train models faster and more efficiently, mainly when dealing with large-scale data sets. However, it raises data privacy concerns and requires additional resources for data management and coordination. So, maintaining data security is always a critical research problem to solve, and research in this area has been accelerated significantly with the advent of federated learning.[3–5] With the rapid increase in computational capacity of edge devices[6–8] and advancements in efficient communication technologies with low power consumption,[9] FL is being applied nowadays in many fields.[10,11]

The concept of federated machine learning involves collaborative machine learning model building across multiple workers while ensuring data privacy, where the raw data remains locally across numerous local devices. Despite recent progress in federated learning on privacy preservation, FL is still vulnerable to poisoning attacks, e.g., data poisoning attacks. In data poisoning attacks, a malicious FL participant alters their training data by adding poison to the instances or changing existing instances in an adversarial manner. Moreover, attackers can dominate a certain number of participants and inject poisonous data directly into the model's training data to degrade the model's performance. Since the server has no access to the training data in FL, there is no authority to validate the data, and these malicious data can poison the global model.

Further author information: (Send correspondence to Pretom Roy Ovi)
Pretom Roy Ovi: E-mail: povi1@umbc.edu
Aryya Gangopadhyay: E-mail: gangopad@umbc.edu
Robert F. Erbacher: E-mail: Robert.F.Erbacher.civ@army.mil
Carl Busart: E-mail: carl.e.busart.civ@army.mil

Existing prevention methods of data poisoning attacks are primarily designed for centralized data collection scenarios, and their effectiveness in federated learning settings needs further investigation. Therefore, there is a need for further research to make federated learning more resistant to data poisoning attacks while preserving data privacy. Authors[12] pointed out that dirty-label data poisoning attacks tend to cause a lot of misclassifications, up to 90%, when an attacker adds a small number of dirty-label samples to the training dataset. And authors[13] pointed out the data poisoning attacks in FL as an issue that needs immediate addressing. This severe issue can compromise the accuracy and reliability of the machine learning model.

To address these issues mentioned above, we first focus on the vulnerability of FL systems to malicious participants who aim to poison the global model and attempt to analyze the effect of data poisoning attacks in federated training. We make minimal assumptions about the capabilities of these malicious participants, assuming that they can only manipulate the raw training data on their local devices. It allows even non-expert attackers to carry out data poisoning attacks without knowing the model type, parameters, or FL process. Under this set of assumptions, label-flipping attacks become a feasible strategy to implement data poisoning attacks which are already shown to be effective against traditional, centralized ML models.[14] We conducted experiments to demonstrate poisoning attacks on MNIST, Fashion-MNIST, and CIFAR-10, widely used image classification datasets. Our findings reveal that the effectiveness of the attack, measured by the decrease in model utility, depends on the percentage of malicious users involved. Even when only a small percentage of users are malicious, the attack can still be effective. Additionally, we found that the attacks can be targeted, meaning they have a significant negative impact on a subset of classes that are under attack while having little to no impact on the remaining classes. This is advantageous for adversaries who want to poison only a specific subset of classes while avoiding the detection of a completely corrupted global model. Finally, we utilize the concept of confident learning[15] to validate the label quality of the data and propose a confident federated learning (CFL) framework to prevent data poisoning attacks. We also demonstrate its potential in detecting the poisonous training samples and refraining them from the local training in FL. Our approach typically involves estimating label noise probabilities or using a confidence threshold to identify potentially mislabeled instances and then preventing them from the local training on the worker sides. The major contributions we have included in this work are:

- *Effect of Compromised Clients on Performance:* In the federated training setup, we first showed how compromised clients under data poisoning attacks affect the performance of the global model. Experimental results suggested that data poisoning attacks could be targeted, and even a small number of malicious clients cause the global model to misclassify instances belonging to targeted classes while other classes remain relatively intact.

- *Prevention of Data Poisoning Attacks:* We propose a confident federated learning (CFL) framework to prevent data poisoning attacks during local training on the worker sides. Along with ensuring data privacy and confidentiality, our proposed approach can successfully detect mislabeled samples and prevent them from local training.

The remainder of this work is organized as follows. First, section 2 provides a brief review of related works. Next, section 3 introduces the proposed methodology to tackle data poisoning attacks. Then, we report the implementation details in section IV and finally discuss the limitation and potential directions in the conclusion.

2. LITERATURE REVIEW

In this section, we will go over the previous works on the adversarial attacks in federated learning.

By their nature, deep learning models necessitate a substantial amount of data to make good predictions. This specific requirement gives rise to a potential problem of the data breach. Moreover, numerous adversarial attacks have been launched to target machine learning and deep learning models. However, most research has focused on poisoning the models in the traditional setting, where a centralized party collects the training data. So, many of the attacks and defenses designed for traditional machine learning do not apply to FL because the server only observes local updates from FL participants, not their instances.

The growing usage of federated learning has prompted research into different types of attacks, such as backdoor attacks,[16,17] gradient leakage attacks,[18–20] poisoning attacks[21–23] and membership inference attacks.[24] The type of attack that is particularly relevant to our research is poisoning attacks, which can be classified into two categories: model poisoning and data poisoning. Model poisoning can be effective in specific scenarios, whereas data poisoning may be preferable due to the fact that it does not require malicious manipulation of the model learning software on participant devices, making it efficient and accessible for non-expert poisoning participants. Our research explicitly investigates data poisoning attacks in the context of federated learning. Authors[12] launched targeted backdoor attacks using data poisoning and pointed out that using only a small number of poisonous samples achieves an attack success rate of above 90%. Data poisoning attacks can cause substantial drops in classification.[21] In data poisoning attacks,[21] the adversary can introduce a number of data samples it wishes to miss-classify with the desired target label into the training set. This data-centric poisoning can be two types- label-flipping[25] and dirty labeling. An attacker can easily manipulate its local data by directly swapping the labels of honest training instances of one class (the source class) to a specific target class while keeping the features of the training data unchanged. It is known as the label-flipping attacks,[26] which can cause significant drops in the performance of the global model even with a small percentage of malicious clients.[21] And in dirty labeling, the attacker aims to add some out-of-box samples (irrelevant samples for training) to the training dataset and mislabels them. For example, the server wants the global model to be trained on MNIST data, but the attacker adds some samples on the training set which are not even the digit (e.g., animal images but label them as digit class). This type of labeling is known as dirty labeling.

Several methods have been suggested to address poisoning attacks in FL.[27–30] These defense strategies are designed to identify malicious updates that deviate from benign updates. For instance, Auror[27] attempts to cluster model updates into two classes, in which the smaller class will be identified as the malicious class and filtered out. Furthermore, the author introduces Byzantine-robust FL methods: Krum,[28] in which the client with the smallest sum of distances to other clients will be selected as the global model. However, in the presence of a large number of malicious participants, these approaches were found to be ineffective in defending against the attacks. Moreover, these defenses are designed for model poisoning attacks. In contrast, in respect to data poisoning attacks, authors[22] proposed a class-wise cluster-based representation to detect the malicious participants attacked by data poisoning, and some other researchers[12,23] pointed out the effect of data poisoning on the performance of the global model. However, the research gap lies in the prevention strategy of data poisoning attacks that can effectively prevent such attacks.

3. METHODOLOGY

In this section, we describe the detailed architecture and work flow of our proposed federated training. Our proposed framework is built on top of the FedAvg[3] algorithm where some of the notations used in this description are $\mathcal{N} = \{1, \ldots, N\}$ signify the set of N clients, each of which has its own dataset $D_{k\in\mathcal{N}}$. Each client trains a local model on its local dataset and only shares the weight updates of local model with the server. Then, the global model, $\mathbf{w}_G$ formation takes place with all the local model updates which can denoted by $\mathbf{w} = \cup_{k\in\mathcal{N}}\mathbf{w}_k$. The proposed federated framework is depicted in figure 1. The process based on the workload of server and client is described below:

1. **Executed in server**

   - *Weight initialization:* The server determines the type of application and how the user will be trained. Based on the application, the global model is built in the server. The server then distributes the global model $\mathbf{w}_G^0$ to selected clients.
   - *Aggregation and global update:* The server aggregates the local model updates from the participants and then sends the updated global model $\mathbf{w}_G^{t+1}$ back to the clients. The server wants to minimize the global loss function[13] $L\left(\mathbf{w}_G^t\right)$, i.e.

$$L\left(\mathbf{w}_G^t\right) = \frac{1}{N}\sum_{k=1}^{N} L\left(\mathbf{w}_k^t\right) \tag{1}$$

This process is repeated until the global loss function converges or a desirable training accuracy is achieved. The Global Updater function runs on the SGD[31] formula for weight update. The formal equation of global loss minimization formula by the averaging aggregation at the t^{th} iteration is given below:

$$\mathbf{w}_G^t = \frac{1}{\sum_{k\in\mathcal{N}} D_k} \sum_{k=1}^{N} D_k \mathbf{w}_i^t \tag{2}$$

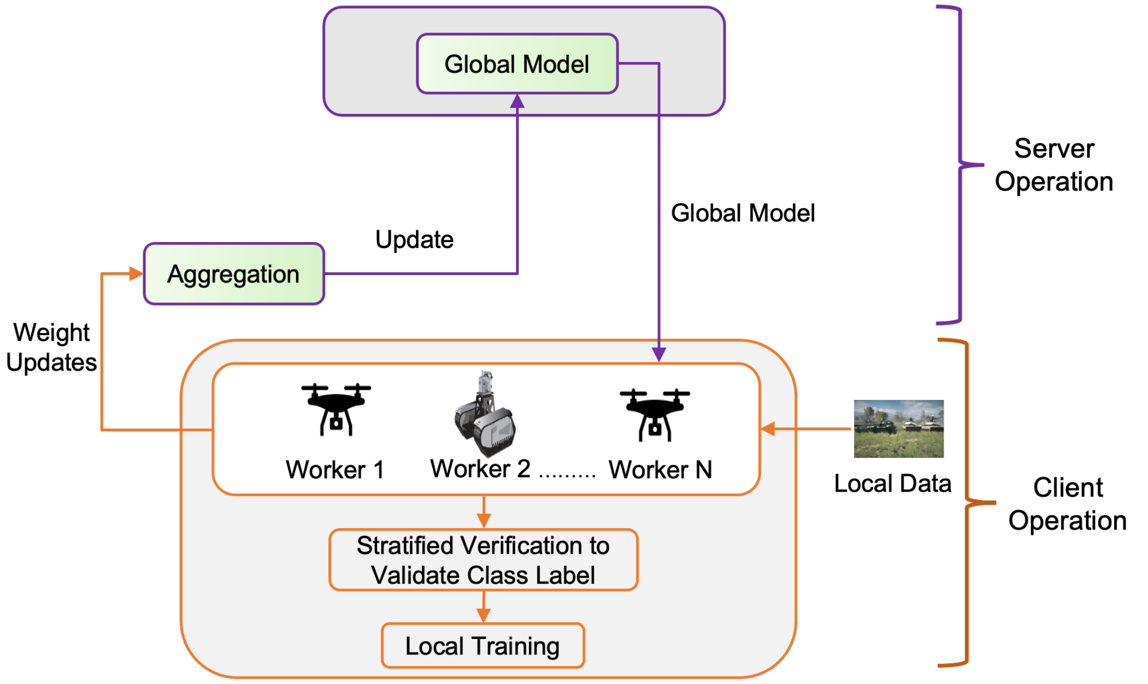


Figure 1: Proposed Federated Framework

2. **Executed at the client level**

- *Validate label quality:* Each client will first validate the label quality of local dataset after receiving the global model $\mathbf{w}_G^0$ from the server. To detect the mislabeled samples in local dataset, every participant needs to compute the out-of-sample predictions for every data point of its local dataset. Then out-of-sample prediction and given labels will be utilized to estimate the joint distribution of given and latent true labels, finally estimate the label errors for each pair of true and noisy classes, details in 3.1. After validating the class labels of every datapoint in local dataset, overall health score for local dataset can be calculated to track label quality.
- *Local training:* Each client will utilize $\mathbf{w}_G^t$ from the server, where t stands for each iteration index, start local training on verified local training samples where detected mislabeled samples are discarded from the local training. The client tries to minimize the loss function[13] $L\left(\mathbf{w}_k^t\right)$ and searches for optimal parameters $\mathbf{w}_k^t$.

$$\mathbf{w}_k^{t^*} = \arg\min_{\mathbf{w}_k^t} L\left(\mathbf{w}_k^t\right) \tag{3}$$

- *Local updates transmission:* After each round of training, updated local model weights are sent to the server afterwards. In addition, each client may send the health score of local data so that the server may monitor the label validation process by tracking the health score.

3.1 Stratified Training to Validate Label Quality

To identify label errors across the local dataset, out-of-sample predicted probabilities for each data point needs to be computed first with K-fold cross-validation. Out-of-sample predicted probabilities refer to the model's probabilistic predictions made only on data points not shown to the model during training. For example, in a traditional train-test split of the data, predicted probabilities generated for the test set can thus be considered out-of-sample. K-fold cross-validation can be used to generate out-of-sample predicted probabilities for every data point in the training set.

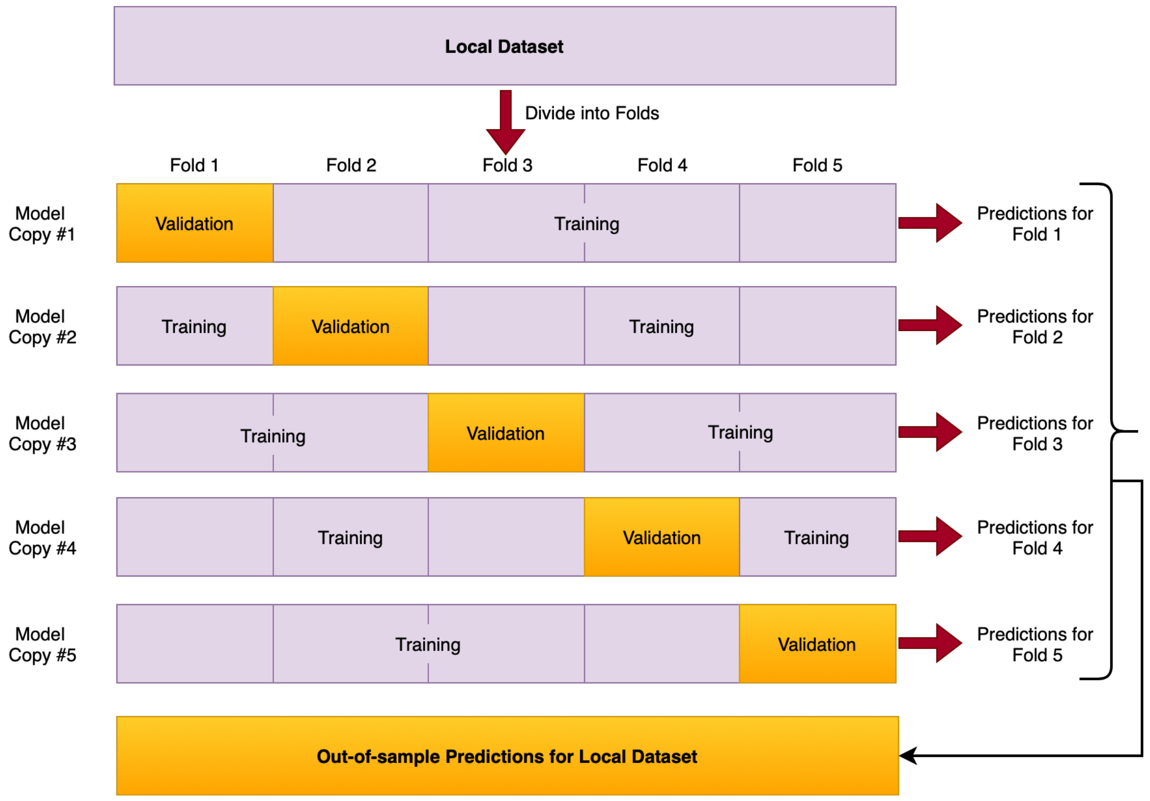


Figure 2: Out-of-sample predictions for local dataset with K-fold cross validation

The diagram in figure 2 depicts K-fold cross-validation with K = 5. K independent copies of our model are trained, where for each model copy, one fold of the data is held out from its training (the data in this fold may be viewed as a validation set for this copy of the model). Each copy of the model has a different validation set for which we can obtain out-of-sample predicted probabilities from this copy of the model. Since each data point is held-out from one copy of the model, this process allows us to get out-of-sample predicted probabilities for every data point. We recommend to apply stratified cross-validation, which tries to ensure the proportions of data from each class match across different folds. Then the out-of-sample predicted probabilities and noisy (given) labels will be used to estimate the joint distribution of noisy (given) and true labels, and find the incorrectly labeled samples in the local dataset.

$$t_j = \frac{1}{|\boldsymbol{X}_{\tilde{y}=j}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=j}} \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \tag{4}$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\} \tag{5}$$

Here, $\tilde{y}$ is discrete random variable takes a given noisy label, y^* is discrete random variable takes the unknown, true latent label, the threshold t_j is the expected (average) self-confidence for each class, $\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}$ is the set of examples x labeled $\tilde{y} = i$ with large enough $\hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta})$ to likely belong to class $y^* = j$, determined by a per-class threshold t_j. And $C_{\tilde{y},y^*}$ is the confident joint formally defined as eqn 6. The confident joint $C_{\tilde{y},y^*}$ estimates $\boldsymbol{X}_{\tilde{y}=i,y^*=j}$, the set of examples with noisy label i that actually should have true label j, by partitioning X into estimate bins $\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}$. When $\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} = \boldsymbol{X}_{\tilde{y}=i,y^*=j}$, then $C_{\tilde{y},y^*}$ exactly finds label errors.

$$\begin{aligned} &\boldsymbol{C}_{\tilde{y},y^*}[i][j] := \mid \hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} \mid \text{ where} \\ &\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} := \left\{ \boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j, j = \arg\max_{l \in [m]: \hat{p}(\tilde{y}=l; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_l} \hat{p}(\tilde{y}=l; \boldsymbol{x}, \boldsymbol{\theta}) \right\} \end{aligned} \tag{6}$$

4. EXPERIMENT SETUP AND RESULT ANALYSIS

This section will step through the detailed experiment setup and result analysis.

4.1 Implementation Details

Federated Training Setup: Experiments were conducted on a system with an Intel(R) Core(TM) i9-11900K CPU (8 cores) and an NVIDIA GeForce RTX 3090 GPU. The federated learning framework was implemented using Keras with a TensorFlow backend. The server controlled the training pace, including the number of epochs per round and the overall number of rounds. The server sets the pace of the training, determines the number of epochs per round, and how many rounds of overall training are to be conducted. We utilized the LeNet-5 CNN architecture as a global model and simulated FL training with 15 clients. For each round, 1 epoch of local training is conducted on the client side. We used SGD optimizer and set the learning rate η to 0.01 during training. And the datasets we used in this experiment are pre-divided into training and testing subset. The test set is kept on the sever and used for model evaluation only and is therefore not included in any participants' local train data.

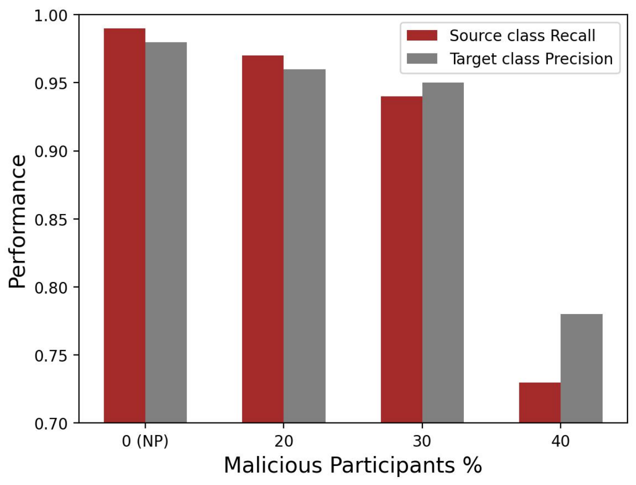
Datasets: To demostrate the attack effect and evaluate the performance of our proposed approach, we conducted experiments on MNIST, Fashion-MNIST, and CIFAR-10 datasets. MNIST and Fashion-MNIST consist of 60000 gray-scale images for training and 10000 for test set associated with 10 classes whereas CIFAR-10 consists of 50000 colored images for training and 10000 for test set.

Label Flipping Process: We randomly choose N x $m\%$ of the participants as malicious to explore the impact of label flipping attack on federated learning system containing N participants, where a particular percentage ($m\%$) of them are malicious. The remaining participants are considered honest. To account for the impact of the random selection of malicious participants, we repeat the experiment multiple times and report the average results. We explore label-flipping attack scenarios for the MNIST, Fashion-MNIST, and CIFAR-10 datasets by flipping the label of the source class to a specific target class denoted as source class $\rightarrow$ target class pairing. For MNIST, we experiment with the following pairings: 0: digit_0 $\rightarrow$ 2: digit_2, 1: digit_1 $\rightarrow$ 5: digit_5, and for Fashion-MNIST, we evaluated the pairing on 1: trouser $\rightarrow$ 3: dress, 0: t-shirt/top $\rightarrow$ 4: coat. For CIFAR-10, we experiment with the following pairings: 6: shirt $\rightarrow$ 7: t-shirt/top, 1: trouser $\rightarrow$ 3: dress.

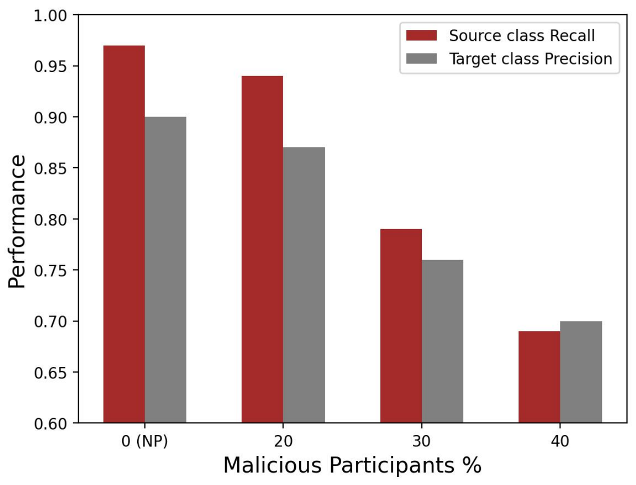
4.2 Result Analysis

Attack Effects Analysis: Label flipping denoted by src $\rightarrow$ target class indicates that ground truth labels of source class samples' have been flipped with the target class label. If we train a machine learning model on a dataset in which the ground-truth labels of some examples from the source class have been flipped with those of the target class, some samples from the source class will be predicted as belonging to the target class during testing. This implies that the source class will have some false negatives, which will ultimately impact its Recall. On the other hand, the target class will have some false positives, which will affect its Precision. Label-flipping attacks are targeted attacks, meaning they have a significant negative impact on a subset of classes that are under attack while having no impact on the remaining classes. So, to evaluate the attack effects with varying percentages of malicious participants, we utilize Recall for the source class and Precision for the target class as evaluation metrics.

The figure 3 illustrates the degradation of the recall of the source class and precision of the target class when the percentage of malicious participants (m) varies from 0% to 40%. The value 0% signifies that none of the



(a) Impact of attack on MNIST data



(b) Impact of attack on Fashion-MNIST data

Figure 3: Evaluation of attack feasibility and impact of malicious participants on performance.

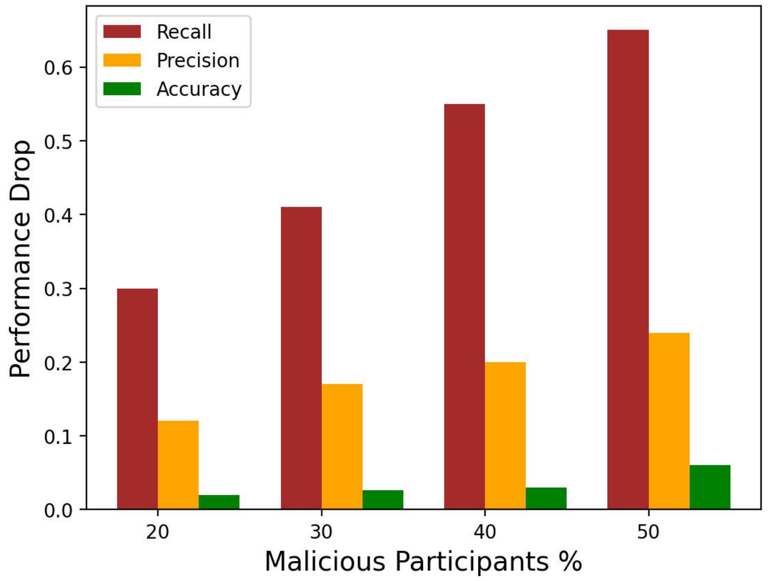
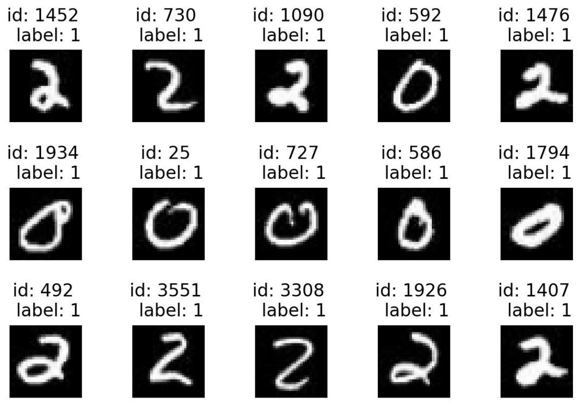


Figure 4: Impact of malicious participants on the performance for CIFAR-10 data.

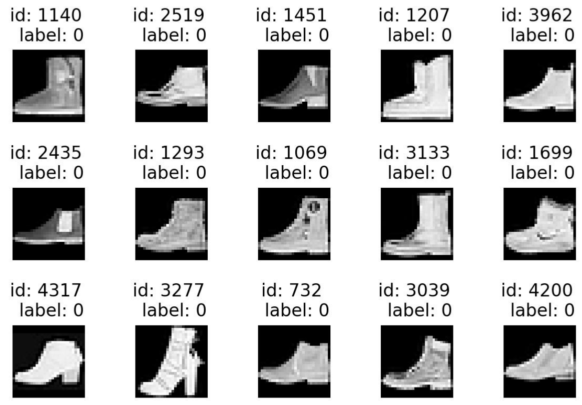
participants are malicious, and therefore, there is no poisoning (NP) in the considered scenario. The findings reveal that with the increase of malicious participants, the global model's potential to predict the source and target classes declines. Even with a small percentage of malicious participants, both the source class recall and the target class precision deteriorate in comparison to a non-poisoned scenario (NP) for both MNIST and Fashion-MNIST, depicted in figure 3a and figure 3b respectively. For example, when malicious participants reach 40%, recall drops from 0.99 to 0.73, and precision drops from 0.98 to 0.78 for MNIST. A similar trend is observed for the Fashion-MNIST dataset as well.

Furthermore, in figure 4, we depicted the global model's performance drop (accuracy, source class recall, and target class precision) for CIFAR-10 data with the increasing percentage of malicious participants. It is clear from the figure that the higher the percentage of malicious participants, the more the performance drop can be. In the experiments, once the malicious percentage reaches 50%, the precision and recall of the targeted classes drop around 0.24 and 0.65, respectively, while the overall accuracy of the global model declines only 0.06.

This result indicates the targeted nature of label-flipping attacks where the attack degrades the global model's potential in predicting the instances belonging to source and target classes while performance for other classes remains relatively the same. And so, the overall accuracy of the global model deteriorates less in comparison with the drop in precision and recall of target and source class, respectively. Therefore, it is evident that an attacker who controls even a small proportion of the total participants can significantly affect the global model's utility.



(a) Detected flipped samples on altered MNIST



(b) Detected flipped samples on altered Fashion-MNIST

Figure 5: Detection of label flipped samples.

Performance Evaluation of Proposed Approach: Based on the analysis presented above, it is clear that preventing data poisoning attacks is crucial in the context of federated training. Our proposed FL framework involves validating the local dataset of each client to identify label-flipped samples prior to the commencement of local training. To evaluate the effectiveness of our approach for detecting mislabeled training samples and preventing them from the local training process, we conducted experiments wherein we deliberately manipulated the local dataset of a few clients to initiate data poisoning attacks. Specifically, we altered the class labels of a few samples in the MNIST dataset (digits 0 and digit 2 images were changed to class label 1). Similarly, in Fashion-MNIST, t-shirt/top images are assigned to class label 0, and ankle boot images are assigned to class label 9. We intentionally altered the label of some ankle boot samples to class label 0. Experimental results using both datasets demonstrate that our proposed approach can successfully detect the poisoned samples of local workers, shown in figure 5, with an accuracy over 88% for MNIST and 86% for Fashion-MNIST, albeit with a small number of false positives (approximately 5%).

5. CONCLUSION

In this paper, we first study the feasibility of data poisoning attacks in federated learning and the negative impact of such attacks on the performance of a global trained model. As the number of malicious participants increases, we observe a decline in the precision and recall of targeted classes, as well as the overall accuracy of the global model. Finally, we evaluate the effectiveness of the proposed method in detecting the poisonous samples and preventing them from local training to prevent attacks in federated learning. In FL, each worker is restricted from accessing the local data of others, but they do have access to their own local data. And our approach can detect the incorrectly labeled data by learning with correctly labeled data from the local dataset. So, if incorrectly labeled samples constitute the majority and/or the local training dataset of a worker is entirely poisoned, our prevention strategy will fail to prevent attacks for the particular worker. In such circumstances, detection methods for data poisoning attacks should come into action to detect malicious participants. To conclude, our results highlight the efficacy of our proposed FL framework in detecting and preventing incorrectly labeled data from being included in local training, thus mitigating the risks associated with data poisoning attacks in federated learning.

ACKNOWLEDGMENTS

This research is supported by U.S. Army Grant No. W911NF21-20076.

REFERENCES

[1] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., et al., "Large scale distributed deep networks," *Advances in neural information processing systems* **25** (2012).

[2] Duan, M., Li, K., Liao, X., and Li, K., "A parallel multiclassification algorithm for big data using an extreme learning machine," *IEEE transactions on neural networks and learning systems* **29**(6), 2337–2351 (2017).

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

[4] Konečný, J., McMahan, B., and Ramage, D., "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575* (2015).

[5] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečný, J., Mazzocchi, S., McMahan, B., et al., "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems* **1**, 374–388 (2019).

[6] Ovi, P. R., Dey, E., Roy, N., and Gangopadhyay, A., "Aris: A real time edge computed accident risk inference system," in [*2021 IEEE International Conference on Smart Computing (SMARTCOMP)*], 47–54, IEEE (2021).

[7] Rashid, H.-A., Ovi, P. R., Gangopadhyay, A., and Mohsenin, T., "Tinym2net: A flexible system algorithm co-designed multimodal learning framework for tiny devices," *ArXiv* (2022).

[8] Ahmed, M., Khan, N., Ovi, P. R., Roy, N., Purushotham, S., Gangopadhyay, A., and You, S., "Gadan: Generative adversarial domain adaptation network for debris detection using drone," in [*2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*], 277–282 (2022).

[9] Mahmud, M., Younis, M., Islam, M. S., Choa, F.-S., and Carter, G., "Vapor cloud delayed-dppm modulation technique for nonlinear optoacoustic communication," in [*GLOBECOM 2022 - 2022 IEEE Global Communications Conference*], 819–824 (2022).

[10] Ovi, P. R., Dey, E., Roy, N., Gangopadhyay, A., and Erbacher, R. F., "Towards developing a data security aware federated training framework in multi-modal contested environments," in [*Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications IV*], **12113**, 189–198, SPIE (2022).

[11] Cho, H., Mathur, A., and Kawsar, F., "Flame: Federated learning across multi-device environments," *arXiv preprint arXiv:2202.08922* (2022).

[12] Chen, X., Liu, C., Li, B., Lu, K., and Song, D., "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526* (2017).

[13] Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., and Miao, C., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials* **22**(3), 2031–2063 (2020).

[14] Xiao, H., Xiao, H., and Eckert, C., "Adversarial label flips attack on support vector machines," in [*ECAI 2012*], 870–875, IOS Press (2012).

[15] Northcutt, C., Jiang, L., and Chuang, I., "Confident learning: Estimating uncertainty in dataset labels," *Journal of Artificial Intelligence Research* **70**, 1373–1411 (2021).

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

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

[18] Melis, L., Song, C., De Cristofaro, E., and Shmatikov, V., "Exploiting unintended feature leakage in collaborative learning," in [*2019 IEEE symposium on security and privacy (SP)*], 691–706, IEEE (2019).

[19] Zhu, L., Liu, Z., and Han, S., "Deep leakage from gradients," *Advances in neural information processing systems* **32** (2019).

[20] Ovi, P. R., Dey, E., Roy, N., and Gangopadhyay, A., "Mixed precision quantization to tackle gradient leakage attacks in federated learning," *arXiv preprint arXiv:2210.13457* (2022).

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

[22] Ovi, P. R., Gangopadhyay, A., Erbacher, R. F., and Busart, C., "Secure federated training: Detecting compromised nodes and identifying the type of attacks," in [*2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*], 1115–1120, IEEE (2022).

[23] Sun, G., Cong, Y., Dong, J., Wang, Q., Lyu, L., and Liu, J., "Data poisoning attacks on federated machine learning," *IEEE Internet of Things Journal* **9**(13), 11365–11375 (2021).

[24] Nasr, M., Shokri, R., and Houmansadr, A., "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in [*2019 IEEE symposium on security and privacy (SP)*], 739–753, IEEE (2019).

[25] Biggio, B., Nelson, B., and Laskov, P., "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389* (2012).

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

[27] Shen, S., Tople, S., and Saxena, P., "Auror: Defending against poisoning attacks in collaborative deep learning systems," in [*Proceedings of the 32nd Annual Conference on Computer Security Applications*], 508–519 (2016).

[28] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J., "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems* **30** (2017).

[29] Fung, C., Yoon, C. J., and Beschastnikh, I., "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866* (2018).

[30] Zhang, Z., Cao, X., Jia, J., and Gong, N. Z., "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in [*Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*], 2545–2555 (2022).

[31] Li, Z., Sharma, V., and Mohanty, S. P., "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consumer Electronics Magazine* **9**(3), 8–16 (2020).