Federated Learning Trustworthiness against Label Flipping in Complex Intelligent Environments

Mohammad Ali Zamani
Electrical and Computer Engineering Faculty
University of Tehran
Tehran, Iran
matin.zamani99@ut.ac.ir

Fatemeh Amiri
Department of Computer Engineering
Hamedan University of Technology
Hamedan, Iran
f.amiri@hut.ac.ir

Nasser Yazdani[*]
Electrical and Computer Engineering
University of Tehran
Tehran, Iran
yazdani@ut.ac.ir

Received: 4 March 2025 – Revised: 3 June 2024 - Accepted: 10 July 2024

Abstract— **A decentralized method of machine learning, federated learning (FL) enables several clients to work together to train models without disclosing their raw data. However, because of its openness, it is also susceptible to poisoning assaults, especially label-flipping (LF), in which harmful clients alter training labels to taint the global model. Such effort drifts the model in a way that the model performance dwindles in specific attack-related classes while behaving the same as benign clients for other classes to increase complexity for detecting solutions. We combat this by using a defense mechanism that dynamically modifies trust factors to filter out malicious updates based on last-layer gradient similarity. By assessing the defense across a variety of datasets and more complex adversarial scenarios, such as multi-group attacks with different intensities, this study builds on earlier studies. According to experimental data, the method maintains accuracy within a proper level of the clean model while drastically reducing the impact of label-flipping, cutting the attack success rate by 50%. These results demonstrate how important it is to have adaptive security measures in place to protect FL models in hostile and changing contexts.**

Keywords: Federated Learning, Machine learning robustness, Label-flipping attacks, Targeted poisoning attacks, complex intelligent agents.

Article type: Research Article

I. Introduction

Machine learning (ML) has revolutionized diverse domains such as computer vision, natural language processing (NLP), and recommendation systems [1]. The effectiveness of ML models, especially deep learning-based architectures, strongly relies on the presence of high-quality large-scale datasets. Acquiring such datasets presents major challenges from data privacy, data ownership, and regulatory perspectives [2]. Many organizations, such as those in healthcare, finance, and e-commerce, are reluctant to share data due to legal compliance requirements (e.g., GDPR and HIPAA) and the risk of exposing sensitive user information [3].

However, this approach raises privacy concerns, and to secure the exchange of information between parties, Federated Learning (FL) is proposed since it facilitates multiple parties to jointly train a model

[*] Corresponding Author

without sharing raw data [4]. FL does not centralize the dataset but allows training a model in a distributed manner and sends the global model to involved clients' processors that only send back to the server their model gradients or parameters [5]. Although this decentralized method reduces privacy exposure, it incurs security threats like poisoning attacks that allow malicious clients to adjust the updates maliciously [6]. One of these attacks is the label-flipping attack (LF), which remains a well-known challenge of FL, in which malicious clients flip the label of their training data to poison the global model but keep an acceptable standard behavior [7].

Decentralized machine learning was originally proposed to counteract the limitations of centralized machine learning, where a volume of large data is transferred to a central server to perform the training [8]. In distributed learning, multiple system nodes collaborate by sharing data and updating the model over a distributed system, improving scalability by reducing single system computational burden [9]. Nevertheless, this method involves exchanging raw data among nodes and is subject to privacy risks and security threats [2]. These issues are especially relevant for applications in sensitive contexts such as health care, finance, and mobile computing, where data confidentiality is paramount. Furthermore, information sent backward over a wide area network incurs network bandwidth restrictions and inter-region data movement costs, thereby presenting more problems [10]. Consequently, researchers explored novel methods that would facilitate cooperative analytics without requiring shared private data.

In 2016, Google proposed Federated Learning (FL) as a privacy-oriented alternative to conventional distributed learning [4]. FL, which stands for Federated Learning, differs from some of the previous methodologies in the sense that this allows multiple clients (e.g., mobile devices, IoT sensors, organizations) to collaboratively train a high-quality shared model while keeping this data local. This approach, known as federated learning, avoids sharing sensitive data by allowing each client to train the model guardedly on local data and exchange model updates, avoiding the need to transfer the entire dataset to a central server, which combines the updates to enhance the global model [5]. This decentralized paradigm substantially mitigates the potential for data leakage [5] while simultaneously reducing communication costs, resulting in the applicability of FL in practical settings like personalized AI services, medical studies, and autonomous systems [11]. FL allows for model training with theoretical guarantees on performance and scalability over heterogeneous datasets decentralized across FL nodes, making it a promising alternative to traditional distributed learning approaches [12].

FL, due to its decentralized nature – with clients fully in control of their local data and training process – exposes several crucial security threats. The autonomy given by FL makes it prone to many poisoning attacks, which can be classified as untargeted and targeted attacks. In an untargeted poisoning attack, adversaries do not seek to degrade the model too strongly on a specific task; rather, the objective is to degrade the overall performance of the global model, causing it not to converge (non-convergence) or reducing the accuracy on all tasks [13]. On the contrary, by launching limited injection poison attacks, targeted poisoning attacks aim at manipulating the global model so that the global model misclassifies a few specific inputs, which makes those attacks stealthier and less perceptible [14]. A classic example of this relationship is the label-flipping (LF) attack, in which attackers modify class labels in their local datasets (e.g., flipping "fraudulent" transactions to "legitimate") whilst not changing input features, resulting in erroneous decisions from the model's perspective [15]. Other kinds of threats include backdoor attacks, in which an adversary embeds target patterns in training data intended to generate hidden trigger patterns [16], and model poisoning attacks, where malicious clients inject malicious updates of the model, rather than the data, and attack the training process directly [17]. The likelihood of these types of attacks is heightened in non-i.i.d data setups, where client distributions can diverge sharply, resulting in a loss of effectiveness when maintaining common defenses. Furthermore, Byzantine failures, where a fraction of clients behave arbitrarily due to either faults or adversarial intent, add another layer of complexity to securing FL [13]. As FL is increasingly adopted in healthcare, finance, and smart IoT systems, developing robust security defenses is critical to preventing data manipulation and adversarial exploitation in real-world deployments.

One major security threat that FL encountered is poisoning attacks, which stem from the inherent defect that clients are in full control of their local data and training process. In federated learning, a group of devices is involved in training the model collaboratively while keeping their data local to them, and such attacks aim to corrupt the global model's integrity, availability, or accuracy by submitting harmful updates to the model learning process. FL poisoning attacks can be roughly classified into two types: data poisoning and model poisoning. In data poisoning attacks, the training dataset is corrupted by modifying labels or injecting adversarial samples into the training data. A well-known attack is the label-flipping attack in which the source-class labels are altered to that of the target classes (i.e., a "legitimate" label gets flipped to become "fraudulent") [14]. In contrast, model poisoning attacks are not focused on altering the dataset but modifying the model gradients directly or scaling up malicious updates to modify the global model's decision boundaries [13], [14], [15], [16], [18], [19]. Both attack types can be either targeted — focusing on misleading predictions for specific inputs—or untargeted, which aims to degrade the model ' s overall performance [16], [19], [20]. Furthermore, backdoor attacks provide hidden triggers inside the model, which results in the model misclassifying inputs that include particular patterns while still functioning normally. In non-i.i.d. FL systems, where anomaly detection is more difficult due to heterogeneous data distributions, the severity of these attacks is increased. Because they rely on oversimplified assumptions about client behavior and data distributions, traditional defenses like Multi-Krum, Trimmed Mean, and FoolsGold have demonstrated poor performance in such scenarios [4], [9], [12], [21],

[22]. Recent studies, such as FL-ProtectX [19], have addressed these issues by introducing methods like adaptive client reweighting and gradient similarity analysis that can distinguish adversarial patterns without the need for pre-established attack fingerprints.

The focused attack type in this paper is label-flipping attacks, which are categorized as targeted poisoning attacks in federated learning architectures where malicious clients intentionally mislabel their own local data by flipping the labels of examples from their corresponding correct class to a target class. As an example, they might mislead the global model by flipping the label of a “cat” to “dog” in their local dataset [23]. From the implementation point of view, adversary clients poison their local datasets by altering class labels while keeping input features unchanged, then submit the malicious model updates during the FL training process. The challenges regarding label-flipping detection spread from data distribution complexities (non-i.i.d forms) to keeping up with other benign clients' performance in non-attack-related classes and making the attack subtle via manipulating the gradients in specific layers. More importantly, the LF attacks' criticality stems from their ease of execution and severe consequences: even a small proportion of adversarial clients can impair the model performance and drift it away from its correct training path [24]. Since the FL models have been used widely for various tasks, such as image classification, medical diagnostics, and financial predictions, the development of a robust, attack-tolerant strategy is pivotal to safeguard the FL models' stability against such deceptive manipulations.

In contrast with the latest recent advances that discuss the LFA in a single adversary group scenario, our model combats such attacks even in more sophisticated setups consisting of multiple adversary groups with even a high ratio of attackers. This goal is achieved via performing last layer gradients angular similarity calculations in a serialized manner to detect adversaries and also their correspondent group. The structure of the paper is as follows: Section II reviews existing work in the field, Section III discusses the problem clearly and formulates the goals and metrics needed, Section IV provides complete information about our mitigation method and algorithms, Section V elaborates more about our experiments’ setup conditions, Section VI covers the simulations’ results, and finally, Section VII concludes the paper.

II. Related Works

To have a clear review of recent advancements for counter-attack and mitigation solutions, we discussed the most recent top-notch methodologies as below.

A. *Contra [13]*

Awan *et al*. proposed the Contra as a practical defense mechanism utilizing an adaptive reputation score for each client based on the angular similarity of the whole model updates fed back to the aggregator over multiple training rounds. Contra illustrated promising results in combating both data and model poisoning attacks, even in extreme non-i.i.d data settings. Despite its significant performance in the mentioned tasks, Contra introduced challenges such as high computational overhead, scalability issues, single adversary group assumption, and failure in incremental adversary setups.

B. *FL-Defender [16]*

Jebreel *et al.* proved that by utilizing gradient similarity analysis, particularly on last-layer gradients, and calculating the cosine similarity between these grabbed gradients, the adversaries are detected clearly in an FL setup. The proposed FL-Defender model demonstrated proper performance for both label-flipping and backdoor attacks [6]. Although its resilience against some attacks, its reliance on last-layer gradients may be circumvented by adaptive adversaries introducing subtle perturbations across different layers. In addition to that, there existed a major assumption in this work that the number of malicious clients is less than 80% of the selected clients in the setup. The authors' other distant assumption was that all malicious clients belong to one group with a single attack intention.

C. *LFighter [15]*

Following the FL-Defender methodology, Jebreel *et al.* applied clustering techniques in parallel with last-layer gradient analysis. Specifically targeting the label-flipping attacks without any attention to similar attack paradigms like backdoors. Dynamic extraction of gradients associated with source and target classes to cluster and isolate the malicious clients’ updates. In contrast with its proper robust performance across various datasets and architectures, its effectiveness may diminish against adaptive adversaries employing subtle or incremental label-flipping strategies. Moreover, its performance in cases of encountering a larger number of attackers or dynamic intention scenarios may drastically dwindles.

D. *FedCC [18]*

Later, Jeong *et al.* introduced a method to tackle both kinds of targeted backdoor attacks and untargeted model poisoning attacks. FedCC accomplishes in distinguishing and filtering out malicious updates exploiting Centered Kernel Alignment (CKA) similarity of Penultimate Layer Representations (PLRs). The methodology described in FedCC was quite effective in non-i.i.d settings.

E. *FLDetector [20]*

Following recent advances in combating targeted or untargeted attacks, Zhang *et al.* came up with an addressing methodology for flagging deviations from the normal benign baseline of clients’ updates in order to distinguish the malicious clients. This was achieved by utilizing time-series anomaly detection techniques to predict adversarial behavior. Although FLDetector’s notable contribution is in detecting previously unseen attack strategies without prior knowledge of attack details, reliance on historical patterns may reduce responsiveness to sudden, one-off poisoning attempts, and also, again the single adversary group assumption is another weakness of the FLDetector.

F. *FL-ProtectX [19]*

Following the advances like Contra [13], FL-Defender [16], and LFighter [15] for combating specifically label-flipping attacks in i.i.d and non-i.i.d setups, we introduced FL-ProtectX. This counterattack and mitigation paradigm showed its performance in complex multi-group adversary scenarios in addition to single adversary scenarios while keeping the attack success rate and other performance metrics in a better range in comparison with other defense schemes such as FedAVG [4], FoolsGold [25], Trimmed Median [12], Mkrum [21], Contra [13], FL-Defender [16]. In previous work, FL-ProtectX was tested in experiments limited to a specific CIFAR-10 dataset and with the assumption of only up to 20% adversaries exist among clients.

In this work, we have expanded the experiment domain of FL-ProtectX to extreme, complex, and sophisticated environments containing more adversaries belonging to multiple groups with various different attack intentions and also exploiting broader datasets in order to prove the FL-ProtectX performance under these intensive circumstances.

III. Problem Definition

A. Deep Neural Networks

Deep neural networks are the building blocks for advanced machine learning techniques, including computer vision [26], speech recognition [27], autonomous systems [28]. The sophisticated relationship between input features and target outputs through multiple mathematical operations is formulated as

$$F(x)=\sigma_L(W_L\cdot\sigma_{L-1}(W_{L-1}...\sigma_1(W_1\mathbf{x}+b_1))+b_L) \tag{1}$$

where $\mathbf{x}$ is the input feature vector, $\boldsymbol{W_l}$ and $\boldsymbol{b_l}$ are the weight matrix and bias vector of the l-th layer, and $\sigma_l(\cdot)$ represents the activation function. Model parameters, which are denoted as $\theta=\{W_l,b_l\}_{l=1}^{L}$ are optimized through the Stochastic Gradient Descent (SGD) [29]. The SGD process consists of repetitive loss function gradient calculation that is formulated as below:

$$\theta^{t+1}=\theta^{t}-\eta\nabla_{\theta}\mathrm{L}(\theta^{t}) \tag{2}$$

where η is the learning rate, and the $\nabla_\theta\mathcal{L}(.)$ is the gradient of the loss with respect to model parameters.

B. Federated Learning

Federated learning accelerated the pace of distributed learning by enabling collaborative model training across distributed devices while preserving each device's privacy by keeping data local and just transferring the model to the aggregator server. The FL process begins with the initialization of the server with a global model and feeding this global model to participants. Each client trains this model with its own local data using SGD as described in Eq. 2. Participants' behaviors are formulated like Eq. 3:

$$\mathrm{L}_k(\theta)=\frac{1}{|D_k|}\sum_{(x_i,y_i)\in D_k}\ell(\mathrm{F}\,(\theta;x_i),y_i) \tag{3}$$

where D_k is the local dataset of the client k, and $\ell(\cdot)$ is the loss function for the specified task. Thereafter, in the local training, each participant feeds back its local model to the aggregator server to aggregate with other participants' models and update the global model. The server might follow multiple different rules for aggregation, e.g., FedAvg [4], FedSVRG[30], FedProx [31], etc. The most well-known aggregation paradigm is FedAvg, where the server calculates the global model via the weighted mean of all received models. Eq.4 illustrates this method as well.

$$\theta^{t+1}=\sum_{k=1}^{K}\frac{|D_k|}{\sum_{j=1}^{K}|D_j|}\theta_k^{t+1} \tag{4}$$

where K represents the total number of participating clients. The whole FL mechanism of initialization, local training, aggregation, updating the global model, and feeding back the global model to clients will last till the global model reaches a dedicated level regarding the performance parameters.

C. Label-flipping attack behavior and complex adversary setup

Among various poisoning strategies against FL, the label-flipping attacks, also known as LFA, are particularly effective due to their simplicity and subtlety since they keep other non-attack-related classes untouched. In an LFA, a malicious participant deliberately alters the class labels of some intended data to its own target label and then begins the local training process. From the statistical point of view, the so-called behavior can be formally defined as a malicious client k with dataset $D_k=(x_i,y_i)$ constructs a poisoned dataset D_k' by flipping the labels of selected records as below:

$$D_k^{'}=\{(x_i,c_t):y_i=c_s\}\cup\{(x_i,y_j):y_j\neq c_s\} \tag{5}$$

where c_s is the source class and c_t is the target class. Since the global model aggregates all clients' updates blindly, it cannot handle the difference between benign and malicious updates. The effectiveness of LFAs may vary regarding the data distribution among clients. In addition to that, such adversarial behavior might make the aggregation method defective against LF if the malicious clients act independently or coordinate their attacks over multiple training rounds.

In our FL setting, we assume a cross-entropy loss function with one-hot encoded labels. Given an activation vector o at the final layer, which passes through the softmax function, the predicted probability for class k is computed as:

$$p_k=\frac{e^{o_k}}{\sum_{j=1}^{|\mathrm{C}|}e^{o_j}},k=1,...,|\mathrm{C}| \tag{6}$$

where $|\mathcal{C}|$ is the total number of classes. Following the cross-entropy method, the loss for specific output y, regarding the probabilities from Eq.6 is derived as:

$$\mathrm{L}(y,p)=-\sum_{k=1}^{|\mathrm{C}|}y_k log(p_k) \tag{7}$$

According to Eq.7, $y=(y_1,y_2,\dots,y_{|C|})$ is the one-hot encoded true label and p_k is the confidence score for the k^{th} class. To investigate the model behavior in the case of affection by LF, the gradient of the loss with respect to the output activation o_i of the i-th neuron in the output layer:

$$\delta_i = \frac{\partial L(y,p)}{\partial o_i} = -\sum_{j=1}^{|C|} \frac{\partial L(y,p)}{\partial p_j} \frac{\partial p_j}{\partial o_i} \tag{8}$$

Since SoftMax influences the probability distribution, we express this gradient as:

$$\delta_i = -\frac{\partial L(y,p)}{\partial p_i} \frac{\partial p_i}{\partial o_i} - \sum_{j \neq i} \frac{\partial L(y,p)}{\partial p_j} \frac{\partial p_j}{\partial o_i} \tag{9}$$

By simplifying, we obtain:

$$\delta_i = p_i - y_i$$

This equation enlightens this point that the divergence between the real one-hot encoded label y_i and the predicted probability p_i determines the gradient direction. This gradient value depends on the output label so that δ_i would in the interval [0, 1] when $y_i = 0$, if the neuron belongs to the wrong class, while it will always be in the interval [-1, 0] when $y_i = 1$ in the case of true class neuron.

Adhering to the above equations, the gradient ∇w_i^L for concerning the weights vector connected to the i^{th} neuron in the output layer:

$$\nabla w_i^L = \delta_i a^{L-1} \frac{\partial \sigma_L}{\partial (w_i^L \cdot a^{L-1} + b_i^L)} \tag{10}$$

where a^{L-1} is the activation output of the previous layer. Likewise, for the bias, we have:

$$\nabla b_i^L = \delta_i \frac{\partial \sigma_L}{\partial (w_i^L \cdot a^{L-1} + b_i^L)} \tag{11}$$

From Eq. 10 and Eq. 11, it is obvious that the gradient δ_i has a direct impact on the gradient of the last layer weights and biases; hence, any mitigation method utilizing this gradient would be effective.

A key observation is that there exists a wide and significant deviation between the malicious and benign gradient directions. The more a client contributes with meaningful updates, the higher the trust score acquires. The effect that arises from adversary clients trying to introduce intentional misclassifications is that the errors of malicious clients tend to diminish as the model converges. As a result, neurons responsible for attack-relevant classes experience larger errors compared to irrelevant neurons, leading to distinct gradient patterns. Over time, irrelevant neurons exhibit reduced activity, and noise introduced by adversaries is gradually pruned. Consequently, gradients of neurons in the final layer retain unique adversarial signatures, making them valuable for identifying label-flipping attacks. This characteristic enables defense mechanisms to focus on layer-specific outputs, isolating attack patterns without relying on complete model information.[32].

With this insight about the label-flipping, we can describe LFAs in complex adversary setups. In a simple LFA, all the adversary clients are supposed to be the same in terms of behavior, meaning that they have the same attack intentions with no conflicts together. In a more sophisticated setup like **Error! Reference source not found.** there exist multiple different groups of adversaries and there is no assurance that all malicious clients seek a similar flipping intention. **Error! Reference source not found.** illustrates an FL setup that consists of K clients each have the proper number of images like CIFAR10 dataset. In the mentioned setting, the first group of adversary clients tend to manipulate its local model by flipping the label of Car to Truck, while the second group clients follow another attack strategy that flips only the label of Plane to Dog. As it is illustrated, the selected clients contribute to the training process in round t are initialized with the global model denoted as G_t and feedback their correspondent local final model also denoted as $w_{i,t+1}$ where i is the index of the client. At the end of round $t+1$ the aggregator combines all collected models and updates the global model. Hence, malicious clients from group 1 reduce the global model performance in predictions regarding to class Car and the group 2 drifts the model to irrelevant predictions for class Plane. Keep in mind that both groups do no harmful or defective action on global model performance on non-attack-related classes.

With the increase in malicious clients no matter which group they belong or increase in attack-related classes among regarding the intelligence level of clients, combating LFAs would get more difficult.

IV. COMBATING SOLUTION

According to III.C the attacker tries to minimize the probability vector of the source class ($p_{c_{src}}$) and maximize the probability vector of the target class ($p_{c_{target}}$) in every single training data existing in its own dataset, which belongs to the source class [13], [15], [16], [19], [20]. Also, the gradients of the last layer of each participant local model have the most valuable information about the model behavior with respect to each class.

As it was discussed in previous sections, in a more realistic scenario, the adversaries' operation domain is much more stochastic in terms of divergence, coherence, and population of attack intention. Thus, coming up with a robust addressing solution that can distinguish the count of adversary groups existing among participant clients with respect to their malicious behavior is a crucial need in Federated Learning technologies.

Our combating solution seeks 4 clear goals including: 1) distinguish the clients sending good updates and those who are sending poisoned updates, 2) diagnose how many adversary groups exists in the environment, 3) detect each malicious client's correspondent group, and 4) maintain model accuracy on all classes, particularly on non-attack-related classes. As it was suggested in FL-Defender [15], [16] and also in our previous work, FL-ProtectX [19] the first step toward combating LFAs is last-layer gradient analysis. **Algorithm *1*** illustrates the mitigation procedure's main steps.

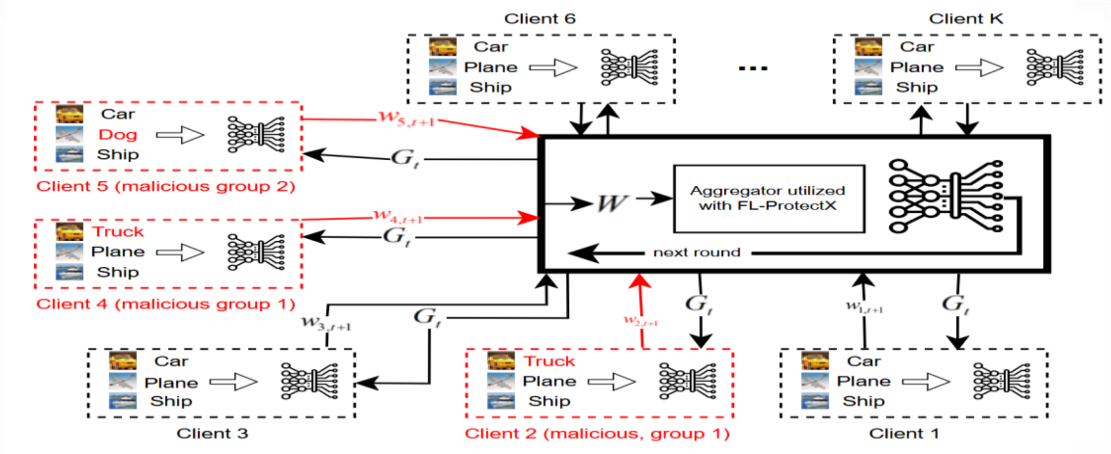


Figure 1. Complex Adversary FL Setup

In each global training round t till the final round T, the defense scheme procedure begins after initializing participant clients, followed by the local training process (**Algorithm *2***), and sending the local models to the aggregator server. Initially, all received local models' last-layer gradients get extracted to calculate the client-wise similarity using the most powerful angular similarity method, cosine similarity[33] formulated as:

$$cs(\nabla_i, \nabla_j) = cos(\theta) = \frac{\nabla_i \cdot \nabla_j}{|\nabla_i||\nabla_j|} \tag{12}$$

Algorithm 1. Whole Mitigation process

```
Algorithm 1 Whole Mitigation process
1: Initialize $W^0$, $H^0 = \{H_k^0 = 0\}_{k=1}^K$;
2: for each round $t \in [0, T-1]$ do
3:     $S \leftarrow$ SELECT PARTICIPANTS RANDOMLY$(C, K)$; m clients are selected randomly to participate in training
4:     INITIALIZE PARTICIPANTS$(W^t$, S)
5:     $W^t \leftarrow$ LOCAL TRAINING PROCESS$(S, W^t)$;
6:     $Sim \leftarrow$ CONSINE SIMILARITY CALCULATION$(S, W^t)$; Sim is exactly the matrix output of cosine similarity function
7:     $\Gamma, Q \leftarrow$ CALCULATE TRUST FACTORS$(Sim, S)$;
8:     //Suppose that the number of adversaries existing in the setup is $f$ (size of Q);
9:     $AGC \leftarrow$ DETECT ADVERSARY GROUPS$(Q)$; //Detect each adversary group correspondent clients and existing groups count
10:    AGGREGATE RESULTS(AGC, W, $\Gamma$); //All results are aggregated using weighted average with respect to $\Gamma$ and AGC
11: end for
```

In comparison with other similarity metrics like Euclidean distance [34], Jaccard [35], Hamming distance [36], or Mahalanobis distance [37] the cosine similarity metric performs by far higher in terms of robustness since this measure only focuses on the angular distance, the critical parameter for finding the separation gap between benign and malicious clients, even if attackers make efforts to increase subtlety by scaling the magnitude of the vector, the measure would produce reliable results. The client-wise cosine similarity calculation procedure is shown in **Algorithm *3***. The mitigation procedure continues with the calculation of trust factors for each client participant in the global training process (**Algorithm *4***).

Algorithm 2. Local Training Process

```
Algorithm 2 Local training process
1: Procedure LOCAL TRAINING PROCESS $(S, W)$
2: for each worker $k \in S$ in parallel do
3:     $W_k^{t+1} \leftarrow$ WORKER_UPDATE$(k, W^t)$; //Each worker $k$ trains $W^t$ using her data $D_k$ locally, and sends her local update $W_k^{t+1}$ back to the aggregator.
4: end for
5: return W
6: End Procedure
```

Algorithm 3. Client-wise Cosine Similarity Calculation

```
Algorithm 3 Cosine similarity calculation
1: Procedure COSINE SIMILARITY CALCULATION $(S, W)$
2: Let $(\nabla_1, ..., \nabla_m)$ be the gradients of the layers of $\{W_k^{t+1} | k \in S\}$;
3: $(cs_{1,1}, ..., cs_{m,m}) \leftarrow$ COSINE SIMILARITY$(\nabla_1, ..., \nabla_m)$; //$cs_{i,j}$ is the cosine similarity between $\nabla_i$ and $\nabla_j$
4: return $(cs_{1,1}, ..., cs_{m,m})$
5: End Procedure
```

Determining the trust factors begins with reducing the dimensions of similarity vectors to two in order to lower the probability of making mistakes due to irrelevant information in the vectors. Then, as a critical technical decision, we used the median of 2-dimensional vectors as the similarity measure since it was supposed to fall in the center of vectors. By considering the gap between the vectors and their median will be clearly distinguished. Each vector distance is the metric for updating the correspondent client's trust factor. Finally, with this first step, all clients' trust factors are calculated and stored as Γ.

Algorithm 4. Trust Factor Calculation

```
Algorithm 4 Calculate Trust factors
Procedure CALCULATE TRUST FACTORS ($Sim$, $S$)
$((p_1^1, p_1^2), ..., (p_m^1, p_m^2)) \leftarrow$ PCA(Sim, $components = 2$);
$CL \leftarrow$ MEDIAN($((p_1^1, p_1^2), ..., (p_m^1, p_m^2))$); //Centroid of compressed similarity vectors selected as the comparison index
for each $(p_i^1, p_i^2)$ do
    $cs_i \leftarrow$ COSINE SIMILARITY($(p_i^1, p_i^2), CL$);
end for
for each worker $k \in S$ do
    $cs_k^t \leftarrow$ Assign($cs_1, ..., cs_m$); //Assign the similarities of the worker to the centeroid
    $H_k^{t+1} \leftarrow H_k^t + cs_k^t$; //Accumulate the similarities of the worker to the centroid to update the history matrix
end for
$Q1 \leftarrow$ COSINE SIMILARITY($H^t$);
$\gamma \leftarrow H^t - Q1$; //Subtract Q1 from every entry in $H^t$. Attackers are expected to be below Q1, thus making their trust values negative.
for each worker $k \in [1, K]$ do
    if $\gamma_k < 0$ then
        $\gamma_k \leftarrow 0$; //Neutralize attacker trust in the aggregation.
    end if
end for
$\Gamma \leftarrow \{\gamma_1, ..., \gamma_K\}$;
$\Gamma \leftarrow \Gamma / max(\Gamma)$;
return $\Gamma, Q1$
End Procedure
```

Algorithm 5. Adversary Detection Procedure

```
Algorithm 5 detect adversary groups
Procedure DETECT ADVERSARY GROUPS ($Q$)
$((p_{a,1}{}^1, p_{a,1}{}^2), ...(p_{a,f}{}^1, p_{a,f}{}^2)) \leftarrow$ GET COMPRESSED SIMILARITY VECTORS($Q$);
$CL_{adv} \leftarrow$ MEDIAN($(p_{a,1}{}^1, p_{a,1}{}^2), ...(p_{a,f}{}^1, p_{a,f}{}^2)$); // Median of adversaries supposed as similarity metric
for each $(p_{a,i}{}^1, p_{a,i}{}^2)$ do
    $cs_{a,i} \leftarrow$ COSINE SIMILARITY($(p_{a,1}{}^1, p_{a,1}{}^2), CL_{adv}$);
end for
$AGC \leftarrow$ FIND UNIQUES($CS_a$);
return AGC
End Procedure
```

The culminating step of the methodology is to determine the count of existing adversary groups in the setup and distinguish their correspondent clients. According to **Algorithm *5***, this is implemented through cosine similarity calculation between all adversary-diagnosed vectors and their median. Finally, the Attacker Group Count metric (AGC) is determined with the interpolation of unique patterns in the final similarity matrix (known as CS_a).

V. EXPERIMENTAL INVESTIGATION

The experiments of this paper are done utilizing the best Python machine learning and AI frameworks: PyTorch and Tensorflow. The computation infrastructure for implementation was RTX 4090 GPU and P100 GPU powered with 200GB RAM disk available on our local supercomputer machine.

In terms of a higher confidence level for generalization of assessing the proposed model assessments, we introduced two different setups. First was the CIFAR10 dataset that includes 60K standard colored images of 10 different classes [38] which was divided into train and test records with the ratio of 50K/10K. In addition to that, we utilized the MNIST dataset that contains 70K handwritten digit images from 0 to 9 [39] which was partitioned into a training set of 60K records and a test set of 10K images.

For the CIFAR10 dataset, the ResNet18 [40] and DenseNet121[41] models have been used as the neural network model for our federated environment global and local models with 11 million and 8 million parameters, respectively. Furthermore, for the MNIST dataset, we used a two-layer CNN with two fully connected layers.

The federated environment setup consists of 20 participants selected randomly, and the dataset is distributed among them equally. The experiments last for 100 global rounds, as each participant trains its model for 3 epochs (local epochs). The batch size for experiments using the CIFAR10 dataset was 32, while for MNIST, the number is 64. Every participant client trained its own model with cross-entropy loss function and used a stochastic gradient descent optimizer, in parallel with 0.01 learning rate and momentum of 0.9. To add no more complexity to the problem, the i.i.d distribution was selected as the sufficient data distribution for this problem. While keeping this condition that each adversary group can follow merely one single flip target at first, i.e., changing one class to another exactly, the number of existing adversary clients rose from 0% to 50% (attacker ratio of 0 to 0.5) of the whole. Noteworthy that adversaries existing in the setting are allocated to each group monotonically.

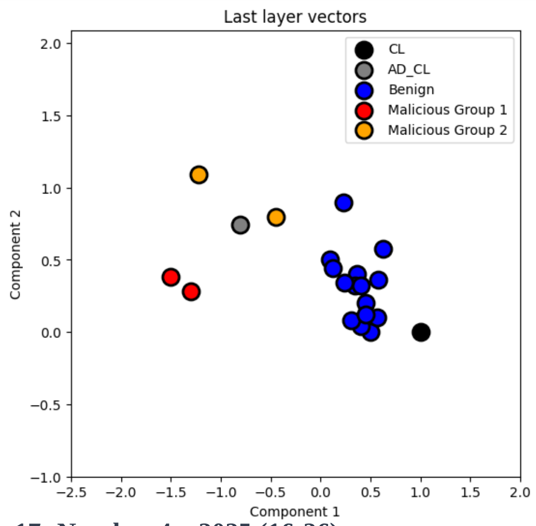
For the whole experiment, we supposed all adversaries belong to two groups only. For the CIFAR10 dataset, the first group flips the label of class 5 (“Dog”) to class 3 (“Cat”), and the second one intends to flip class 6 (“Frog”) to class 4(“Deer”). Similarly, for the MNIST dataset, the primary group tries to flip the label class “9” to class “4,” and the secondary group tries to flip class “7” to class “1”.

To assess further extreme scenarios in a conflict of intentions circumstance, we implemented additional setups for the CIFAR10 dataset using the ResNet18 model. The first group flips the classes from 5 to 3, and the second one makes efforts in a contradictory manner. Also, for the MNIST dataset, the groups exploit the attack by flipping the labels of 4 to 9 and vice versa.

Each experiment performance is evaluated via introducing Test Error (TE): the error resulting from the loss functions used in training, Overall Accuracy (All-Acc): the proportion of correct predictions overall predictions, Source Class Accurcay (Src-Acc): the accuracy for the subset of test examples that belongs to the source class, Attack Success Rate (ASR): the ratio of samples that are unfortunately classified into the adversary desired label, and Adversary Groups Count (AGC): the number of adversary groups detected by the model over the real number of existing adversary groups. Any defense scheme with higher All-Acc and Src-Acc, lower TE and ASR, and predicts the AGC correctly is considered as more successful in combating the LFA.

VI. RESULTS

The novelty of our method in mitigating the LFAs in complex intelligent environments is evaluated in comparison with recent solutions like FL-Defender [16], Trimmed mean (Tmean) [12], multi-Krum (Mkrum) [21], and FedAvg [4]. Noteworthy that none of the mentioned combating solutions make any progress in detecting the existing attacker groups in a complex multi-adversary setup. The mechanism of detecting different adversary groups is illustrated in **Error! Reference source not found.Error! Reference source not found.** There is a crystal-clear gap between benign and malicious clients’ updates, as it is a clear distance among adversaries in terms of angular distance.



Volume 17- Number 4 – 2025 (16-26)

Figure 2. CIFAR10 IID setup gradients

In addition to the last layer gradient vectors analysis, our model performed well in keeping up with other existing mitigation methods and taking them over.

Error! Reference source not found., **Error! Reference source not found.**, **Error! Reference source not found.**, **Error! Reference source not found.**, and **Error! Reference source not found.** provide a clear inspection of the All-Acc parameter for each experiment scenario. It is obvious that in scenarios consisting of a higher rate of malicious adversaries in the environment, the overall accuracy dwindles gradually, which is an inevitable phenomenon. As it can be seen from the comparison of **Error! Reference source not found.** and **Error! Reference source not found.**, the neural network impact can be very effective on the defensive method performance, so when using DenseNet121, the overall accuracy is up to 10% better. In experiments that used the MNIST dataset, the overall accuracy is up to 98.54% in case that half of the clients are malicious, which is a distinguishable result for our model even in such an extreme scenario. Further, in scenarios where the adversaries are in tension as they attack intentions conflict, both for CIFAR10 and MNIST, our model performs properly, showing an overall accuracy of 78% and 98.8%, respectively. The higher accuracy in conflicting scenarios is higher due to that only two classes are involved in the attack rather than four classes in general complex scenarios. Also, the comparison of dataset and model illustrates reasonable and distinguishable symptoms of model impact on the mitigation process, which is laid over the angular analysis of gradients resulting from clients selected for participating in the global learning process. Increasing the number of attackers per group complicates mitigation while amplifying malicious update signals. However, with a proper mitigation method, the larger attacker population could become a disadvantage to adversaries. (Fig. 3 – Fig. 7)

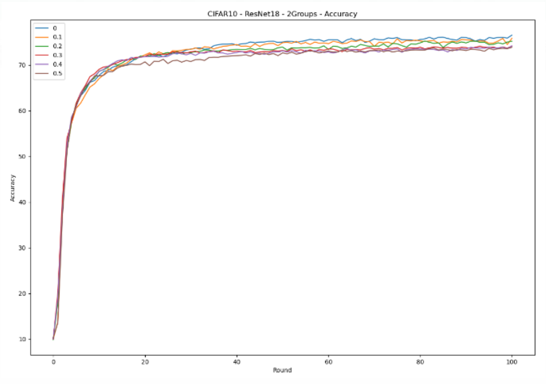


Figure 3. CIFAR10 ResNet18 IID setup Accuracy

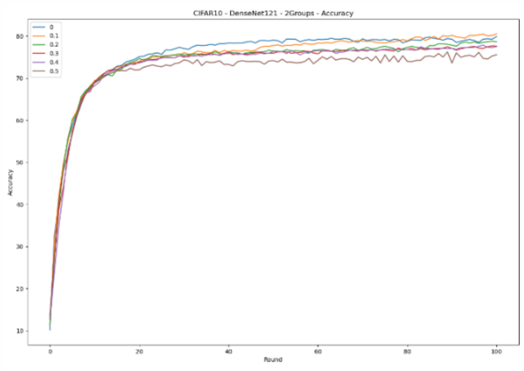


Figure 4. CIFAR10 DenseNet121 IID setup Accuracy

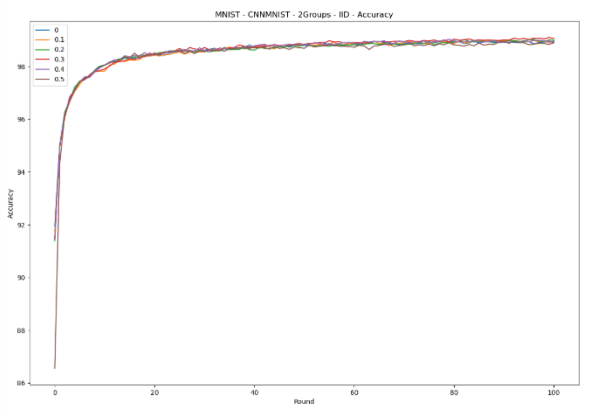


Figure 5. MNIST CNNMNIST IID setup Accuracy

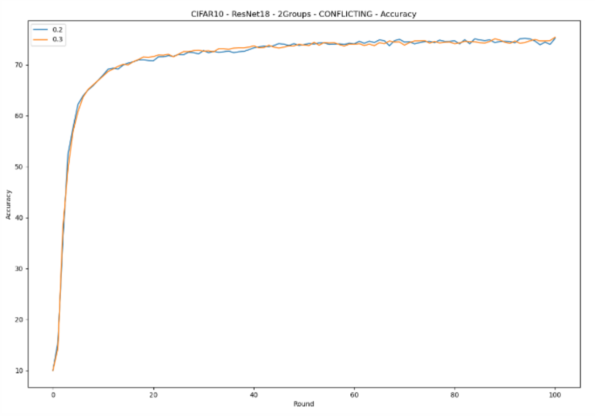


Figure 6. CIFAR10 ResNet18 IID Conflicting setup Accuracy

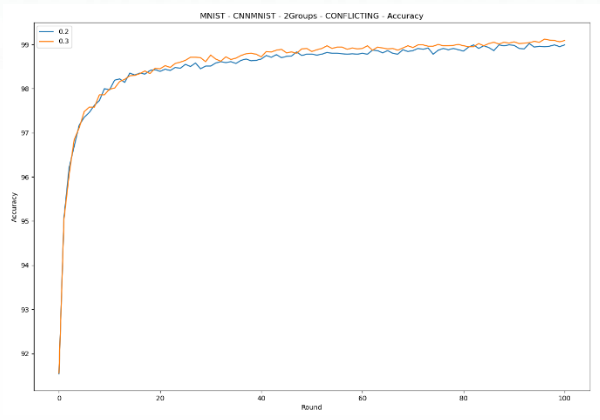


Figure 7. MNIST CNNMNIST IID Conflicting setup Accuracy

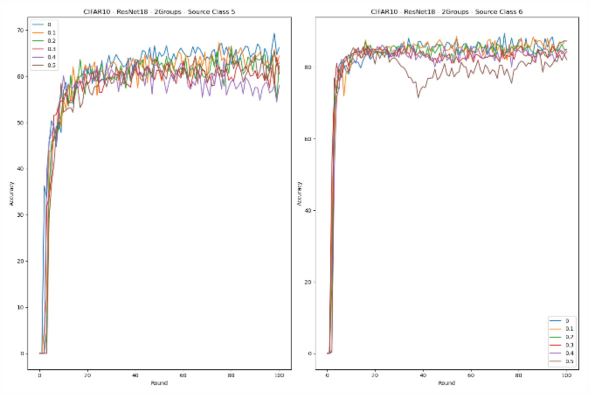


Figure 8. CIFAR10 ResNet18 IID setup Source Class Accuracy

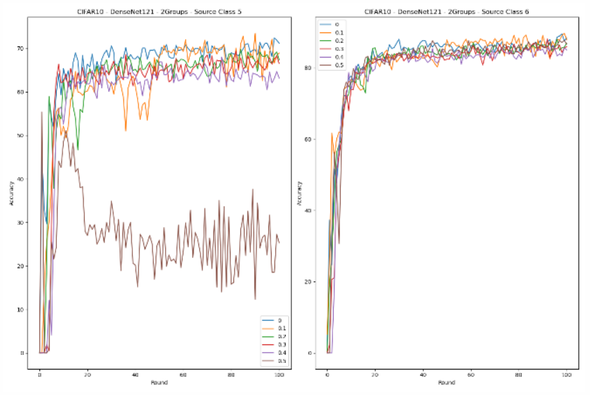


Figure 9. CIFAR10 DenseNet121 IID setup Source Class Accuracy

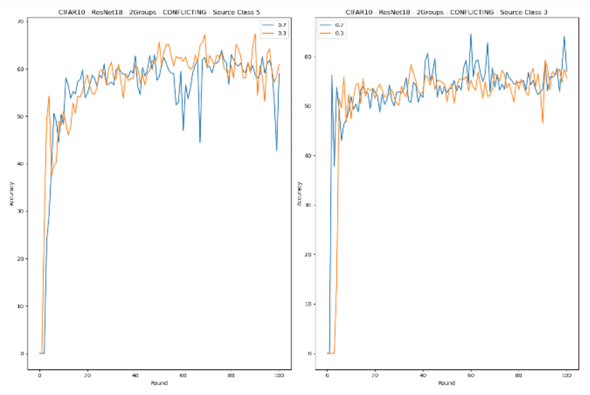


Figure 10. CIFAR10 ResNet18 IID Conflicting Source Class Accuracy

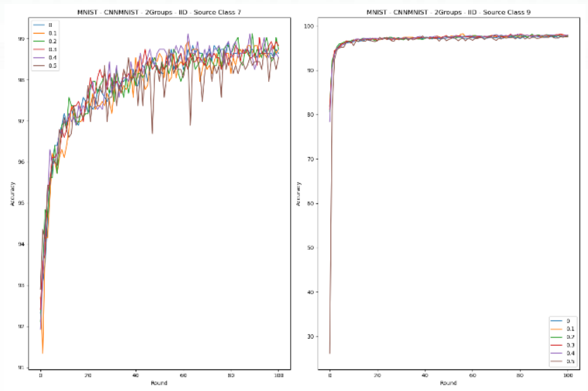


Figure 11. MNIST CNNMNIST IID setup Source Class Accuracy

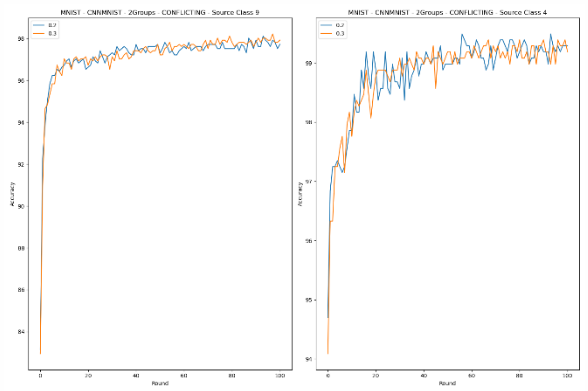


Figure 12. MNIST CNNMNIST IID Conflicting setup Source Class Accuracy

Another important parameter for our experiments is source class accuracy during the operation of malicious clients. In **Error! Reference source not found.**, **Error! Reference source not found.**, **Error! Reference source not found.**, **Error! Reference source not found.**, and **Error! Reference source not found.** the source class accuracies are plotted concerning the correspondent scenario source class label. In both the CIFAR10-ResNet18 and CIFAR10-DenseNet121 setup, class 5 ("Dog") accuracy is around 10% less than class 6 ("Frog"). However, even in such case, the accuracy of the model is not negligent, and our solution demonstrates its power by keeping the source class accuracy level. The main reason for such difference between classes is described with confounding and common features of the source and target class; this means that we may utilize a feature separation indexing algorithm beside our mitigation method. Another proof for our model's significant impact on lowering the impact of LFAs is that for MNIST dataset source class accuracy is 98.54% and 97.52% for class "7" and class "9", respectively**Error! Reference source not found.**. According to **Error! Reference source not found.**, the source class accuracy is even high in a conflict-of-interest setting.

TABLE I. COMPARISON OF COMBATING METHODS IN MULTI-ADVERSARY SCENARIO (CIFAR10)

Benchmark (attacker ratio)	Method	FedAvg [4]	Mkrum [21]	TMean [12]	FI-Defender [16]	Ours

0	TE	0.80	0.83	0.81	0.81	0.81
	All-Acc%	76.90	76.83	**76.91**	76.81	76.82
	Src-Acc%	63.6	65.70	66.50	**67.60**	**67.60**
	ASR	15.60	14.30	13.50	**13.10**	13.15
	AGC	0	0	0	1	**2**
0.1	TE	0.80	0.84	**0.79**	0.82	0.82
	All-Acc%	76.96	**77.04**	**77.04**	76.94	77.00
	Src-Acc%	55.20	53.90	57.90	**65.70**	**65.65**
	ASR	23.40	16.50	24.00	12.00	**11.97**
	AGC	0	0	0	1	**2**
0.2	TE	0.85	0.83	0.81	0.82	**0.80**
	All-Acc%	75.27	75.20	75.76	**76.10**	76.03
	Src-Acc%	44.00	38.90	47.00	**65.10**	66.40
	ASR	33.60	14.60	31.00	15.00	**13.48**
	AGC	0	0	0	1	**2**
0.3	TE	0.9	0.85	0.91	0.85	**0.813**
	All-Acc%	74.38	75.00	72.40	**76.15**	75.04
	Src-Acc%	43.75	38.32	42.39	65.00	**67.93**
	ASR	38.00	18.85	34.83	18.48	14.84
	AGC	0	0	0	1	**2**
0.4	TE	0.85	0.96	0.95	**0.873**	0.9
	All-Acc%	70.32	74.39	70.32	74.33	**76.31**
	Src-Acc%	40.2	38.00	40.12	63.92	**66.33**
	ASR	40.4	20.43	36.00	20.43	**18.32**
	AGC	0	0	0	1	**2**
0.5	TE	1.32	1	1.42	**0.9**	0.903
	All-Acc%	68.4	72.09	64.98	74.01	**74.32**
	Src-Acc%	40	38.50	35	60.95	**64.31**
	ASR	43.07	21.53	40.207	28.35	**21.36**
	AGC	0	0	0	1	**2**

Error! Reference source not found. provides shreds of evidence of our model's power in combating label-flipping concerning sophisticated intelligent FL setups up to 50% better versus the recent advances. This table shows that the performance parameters leveled out from the latest solution, FL-Defender, to our model if no adversaries exist, while as the FL setup adversary population increases, our model demonstrates its strength in taking over other solutions in all defined performance parameters.

VII. Conclusion

FL technologies are always susceptible to various issues threatening the model performance via either data or model manipulation. The malicious clients make an effort to exploit modification in model behavior by sending malignant updates to the aggregator server. One of our observations was that all existing label-flipping attack addressing solutions assumed a single group adversary, whereas in the real world, this assumption might be too distant. Hence, the so-called solutions fall short when the experiment configurations are akin to more complicated scenarios. In this paper, we made a further step toward combating label-flipping attacks to provide a more robust solution that not only mitigates the impact of label-flipping attacks but also can detect the group that adversaries belong to. Our methodology accomplished this intricate task by undergoing multiple serialized angular similarity assessments of all the participants' last layer gradients concerning their median, improving the results by 50%. The empirical results demonstrate that the suggested methodology can keep the overall model performance in addition to reducing the side effects of a label-flipping attack.

To discuss further future research directions in this paper, more complicating technical challenges can be described. Tackling such an attack in non-i.i.d distribution setups that make the addressing process more complicated could be assessed in future research. Furthermore, measuring the model performance and suggesting more top-notch solutions for extreme intensive scenarios where adversaries are more intelligent is another research direction. To elaborate more, limiting the adversaries to only two groups is not a promising assumption; hence, adding more intelligence to the clients to either increase their population or swing between different attack intentions adds to the novelty of attackers and shreds lights for deeper inspection in mitigation methods.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[2] R. Shokri and V. Shmatikov, "Privacy-Preserving Deep Learning," in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver Colorado USA: ACM, Oct. 2015, pp. 1310–1321. doi: 10.1145/2810103.2813687.

[3] P. Voigt and A. Von Dem Bussche, The EU General Data Protection Regulation (GDPR). Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-57959-7.

[4] H. B. McMahan, E. Moore, D. Ramage, and S. Hampson, "Communication-Efficient Learning of Deep Networks from Decentralized Data".

[5] K. Bonawitz et al., "Towards Federated Learning at Scale: System Design," Mar. 22, 2019, arXiv: arXiv:1902.01046. doi: 10.48550/arXiv.1902.01046.

[6] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How To Backdoor Federated Learning".

[7] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data Poisoning Attacks Against Federated Learning Systems," Aug. 11, 2020, arXiv: arXiv:2007.08432. doi: 10.48550/arXiv.2007.08432.

[8] J. Dean et al., "Large Scale Distributed Deep Networks".

[9] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A Survey on Distributed Machine Learning," ACM Comput. Surv., vol. 53, no. 2, pp. 1–33, Mar. 2021, doi: 10.1145/3377454.

[10] J. Konečný, B. McMahan, and D. Ramage, "Federated Optimization:Distributed Optimization Beyond the Datacenter," Nov. 11, 2015, arXiv: arXiv:1511.03575. doi: 10.48550/arXiv.1511.03575.

[11] N. Rieke et al., "The future of digital health with federated learning," Npj Digit. Med., vol. 3, no. 1, p. 119, Sep. 2020, doi: 10.1038/s41746-020-00323-1.

[12] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates".

[13] S. Awan, B. Luo, and F. Li, "CONTRA: Defending Against Poisoning Attacks in Federated Learning," in Computer Security – ESORICS 2021, vol. 12972, E. Bertino, H. Shulman, and M. Waidner, Eds., in Lecture Notes in Computer Science, vol. 12972. , Cham: Springer International Publishing, 2021, pp. 455–475. doi: 10.1007/978-3-030-88418-5_22.

[14] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "Defending against the Label-flipping Attack in Federated Learning," Jul. 05, 2022, arXiv: arXiv:2207.01982. doi: 10.48550/arXiv.2207.01982.

[15] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "LFighter: Defending against the label-flipping attack in federated learning," Neural Netw., vol. 170, pp. 111–126, Feb. 2024, doi: 10.1016/j.neunet.2023.11.019.

[16] N. Jebreel and J. Domingo-Ferrer, "FL-Defender: Combating Targeted Attacks in Federated Learning," Jul. 02, 2022, arXiv: arXiv:2207.00872. Accessed: Aug. 25, 2024. [Online]. Available: http://arxiv.org/abs/2207.00872

[17] N. M. Jebreel, J. Domingo-Ferrer, A. Blanco-Justicia, and D. Sanchez, "Enhanced Security and Privacy via Fragmented Federated Learning," IEEE Trans. Neural Netw. Learn. Syst., vol. 35, no. 5, pp. 6703–6717, May 2024, doi: 10.1109/TNNLS.2022.3212627.

[18] H. Jeong, H. Son, S. Lee, J. Hyun, and T.-M. Chung, "FedCC: Robust Federated Learning against Model Poisoning Attacks," 2022, arXiv. doi: 10.48550/ARXIV.2212.01976.

[19] M. A. Zamani, F. Amiri, P. Jamshidi, S. M. Hosseini, and N. Yazdani, "FL-ProtectX: Combating Label flipping in complex multi-agent environments," in 2024 11th International Symposium on Telecommunications (IST), Tehran, Iran, Islamic Republic of: IEEE, Oct. 2024, pp. 589–594. doi: 10.1109/IST64061.2024.10843483.

[20] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "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, Washington DC USA: ACM, Aug. 2022, pp. 2545–2555. doi: 10.1145/3534678.3539231.

[21] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent".

[22] S. Shen, S. Tople, and P. Saxena, "A uror: defending against poisoning attacks in collaborative deep learning systems," in Proceedings of the 32nd Annual Conference on Computer Security Applications, Los Angeles California USA: ACM, Dec. 2016, pp. 508–519. doi: 10.1145/2991079.2991125.

[23] B. Biggio, B. Nelson, and P. Laskov, "Poisoning Attacks against Support Vector Machines," Mar. 25, 2013, arXiv: arXiv:1206.6389. doi: 10.48550/arXiv.1206.6389.

[24] A. Blanco-Justicia, J. Domingo-Ferrer, S. Martínez, D. Sánchez, A. Flanagan, and K. E. Tan, "Achieving security and privacy in federated learning systems: Survey, research challenges and future directions," Eng. Appl. Artif. Intell., vol. 106, p. 104468, Nov. 2021, doi: 10.1016/j.engappai.2021.104468.

[25] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "The Limitations of Federated Learning in Sybil Settings".

[26] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," Comput. Intell. Neurosci., vol. 2018, pp. 1–13, 2018, doi: 10.1155/2018/7068349.

[27] M. A. Anusuya and S. K. Katti, "Speech Recognition by Machine, A Review," 2010, doi: 10.48550/ARXIV.1001.2267.

[28] T. Zhang et al., "Current trends in the development of intelligent unmanned autonomous systems," Front. Inf. Technol. Electron. Eng., vol. 18, no. 1, pp. 68–85, Jan. 2017, doi: 10.1631/FITEE.1601650.

[29] L. Bottou, "Stochastic Gradient Descent Tricks," in Neural Networks: Tricks of the Trade, vol. 7700, G. Montavon, G. B. Orr, and K.-R. Müller, Eds., in Lecture Notes in Computer Science, vol. 7700. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 421–436. doi: 10.1007/978-3-642-35289-8_25.

[30] D. Chen, C. S. Hong, Y. Zha, Y. Zhang, X. Liu, and Z. Han, "FedSVRG Based Communication Efficient Scheme for Federated Learning in MEC Networks," IEEE Trans. Veh. Technol., vol. 70, no. 7, pp. 7300–7304, Jul. 2021, doi: 10.1109/TVT.2021.3089431.

[31] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," Apr. 21, 2020, arXiv: arXiv:1812.06127. doi: 10.48550/arXiv.1812.06127.

[32] A. K. Singh, A. Blanco-Justicia, and J. Domingo-Ferrer, "Fair detection of poisoning attacks in federated learning on non-i.i.d. data," Data Min. Knowl. Discov., vol. 37, no. 5, pp. 1998–2023, Sep. 2023, doi: 10.1007/s10618-022-00912-6.

[33] A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," in 2016 4th International Conference on Cyber and IT Service Management, Bandung, Indonesia: IEEE, Apr. 2016, pp. 1–6. doi: 10.1109/CITSM.2016.7577578.

[34] K. L. Elmore and M. B. Richman, "Euclidean Distance as a Similarity Metric for Principal Component Analysis," Mon. Weather Rev., vol. 129, no. 3, pp. 540–549, Mar. 2001, doi: 10.1175/1520-0493(2001)129<0540:EDAASM>2.0.CO;2.

[35] J. M. Hancock, "Jaccard Distance (Jaccard Index, Jaccard Similarity Coefficient)," in Dictionary of Bioinformatics and Computational Biology, 1st ed., J. M. Hancock and M. J. Zvelebil, Eds., Wiley, 2004. doi: 10.1002/9780471650126.dob0956.

[36] M. Norouzi, D. J. Fleet, and R. Salakhutdinov, "Hamming Distance Metric Learning".

[37] G. J. McLachlan, "Mahalanobis distance," Resonance, vol. 4, no. 6, pp. 20–26, Jun. 1999, doi: 10.1007/BF02834632.

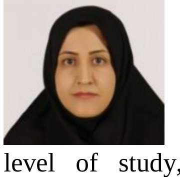
[38] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images".

[39] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object Recognition with Gradient-Based Learning," in Shape, Contour and Grouping in Computer Vision, vol. 1681, in Lecture Notes in Computer Science, vol. 1681. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 319–345. doi: 10.1007/3-540-46805-6_19.

[40] A. V. S. Abhishek, D. V. R. Gurrala, and D. L. Sahoo, "Resnet18 Model With Sequential Layer For Computing Accuracy On Image Classification Dataset," vol. 10, no. 5, 2022.

[41] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI: IEEE, Jul. 2017, pp. 2261–2269. doi: 10.1109/CVPR.2017.243.

Mohammad Ali Zamani holds a Master's degree in Computer Software Engineering from the University of Tehran, with a focus on federated learning security and trustworthy artificial intelligence. He has contributed as a teaching and research assistant in various advanced courses, such as Advanced Algorithms and Advanced Computer Mathematics, at the School of Electrical and Computer Engineering. His research interests span machine learning, optimization, deep reinforcement learning, and secure distributed computing. His work addresses the development of privacy-preserving and efficient models for intelligent networked systems. Currently, he is focused on advancing the security and scalability of federated learning models in multi-agent and complex network environments.



Fatemeh Amiri is currently an assistant professor in the Department of Computer Engineering at Hamadan University of Technology, Iran. She obtained a B.Sc. and M.Sc. in Computer Engineering from Tehran University. To go at a higher level of study, she did her PhD in Computer Engineering between data protection and security which was then completed successfully from an university well-known for its academic merit. Her research areas are machine learning, network security, and sentiment analysis. Dr. Amiri has contributed significantly to both academia and industry. She has also overseen a number of data security and privacy research initiatives and laboratories. Her areas of interest in research are safe computer systems, large data mining, machine learning for cybersecurity, and privacy-preserving data analysis.

Nasser Yazdani is a full professor at the Department of Electrical and Computer Engineering in Tehran University. He got his B.S. degree in Computer Engineering from Sharif University of Technology, Tehran, Iran. He worked in Iran Telecommunication Research Center (ITRC) as a consultant, researcher and developer for few years. To pursue his education, he entered Case Western Reserve Univ, Cleveland, Ohio, USA, later and graduated as a Ph.D. in Computer Science and Engineering. Then, Prof. Yazdani worked in different companies and research institutes in USA. He joined the ECE Dept. of Univ. of Tehran, Tehran, Iran, in Sep. 2000. Prof. Yazdani then initiated different research projects and Labs in high-speed networking and systems. His research interests include networking, packet switching, access methods, distributed systems and database systems.