Journal of Internet Services and Applications, 2026, 17:1, doi: 10.5753/jisa.2026.7098

MONZA: A Score System for Malicious Clients Detection

Rafael Veiga [**Federal University of Pará** | ***rafael.teixeira.silva@icen.ufpa.br***]
Renan Morais [**Federal University of Pará** | ***r299211@dac.unicamp.br***]
Lucas Bastos [**Federal University of the South and Southeast of Pará**| ***bastos.lucas@unifesspa.edu.br***]
Susana Sargento [**Instituto de Telecomunicacões, Universidade de Aveiro** | ***susana@ua.pt***]
Denis Rosário [**Federal University of Pará** | ***denis@ufpa.br***]
Eduardo Cerqueira [**Federal University of Pará** | ***cerqueira@ufpa.br***]

Technology Institute, Federal University of Pará, Av. Perimetral, s/n, Guamá, Belém, PA, 66075-110, Brazil.

Received: 03 December 2025 • **Accepted:** 03 May 2026 • **Published:** 20 July 2026

Abstract Federated Learning (FL) is a machine learning training method that uses a collaborative model for training across a diverse set of clients, while preserving data privacy in accordance with the General Data Protection Regulation (GDPR) for clients' data. The classification and similarity within many clients can become an issue, often leading to decreased model accuracy and slower convergence, or reducing the number of parameters to the point of not learning anything during aggregation. However, the presence of non-IID data and malicious clients poses significant challenges to the significance of generalization models and distribution data. Malicious clients can perform poisoning attacks by sending harmful model updates that degrade the performance of the global model. This article introduces MONZA, a scoring system designed to detect and exclude malicious clients in FL environments. In a scenario where clients can also engage in various attacks, including model poisoning and data poisoning, this can lead to incorrect training. The proposed method uses cosine similarity to calculate client scores. It employs L2 normalization to identify biased models; in some cases, the similarity is not sufficient to classify a client, effectively filtering out malicious participants before aggregation and implementing a penalty with a quarentine method. Our evaluation shows that MONZA achieves an accuracy of 54.5% in a scenario with 30% malicious clients, while zPROBE (i.e., existing resilient methods) only reached an accuracy of 50%. Furthermore, MONZA reduces the simulation execution time by 66% and the computational effort to 83 MFLOP/s compared to zPROBE, which demonstrates to be a more efficient and resilient method. These results confirm that MONZA maintains the integrity of the model, making a security aggregation while minimizing resource consumption in malicious FL settings.

Keywords: Federated Learning, GDPR, non-IID, malicious clients, attacks.

1 Introduction

Integrating big data and deep learning across various applications has significantly improved the intelligence and efficiency of our daily lives Barros *et al*. [2024]. However, the increasing use of traditional Machine Learning (ML) models raises concerns about data privacy, as the transmission of raw data for model training poses privacy risks, particularly when working with large datasets containing sensitive information. In addition, regulatory laws, such as the General Data Protection Regulation (GDPR), have transformed data handling practices worldwide, creating new challenges for data sharing in ML applications Ye *et al*. [2023].

In this context, Federated Learning (FL) addresses these challenges through an innovative method that enables collaborative model training while preserving data privacy. Specifically, rather than sharing raw data as occurs in traditional ML, FL involves clients sharing their locally trained model parameters, which are then aggregated on cloud or edge servers using aggregation policies. This process produces a unified global model that benefits from distributed learning while ensuring that sensitive data remain decentralized between participating clients de Souza *et al*. [2024].

In this context, client selection methods must identify a subset of devices that are eligible for subsequent learning rounds. The success of FL depends on obtaining high-quality, diverse data from all clients and aims at achieving fairer results for comparing clients' models. Therefore, it is important to select the most suitable clients based on the behaviors of their models to improve the convergence of the global model and reduce computational waste in edge processing and accuracy, excluding clients whose data are not helpful for aggregation Yan *et al*. [2023].

However, the aggregation server relies on clients to have correctly updated their models, and therefore malicious clients could exploit the distributed training setup to inject harmful updates Hasan *et al*. [2025]. In this way, different devices have varied data or attempt to harm the training by sending incorrect updates, causing problems with the combination of all the knowledge into a single global model Smestad and Li [2023]. In this sense, malicious clients reduce the accuracy of the model by aggregating the learning from the server with incorrect model updates Yan *et al*. [2023]. For example, as training continues, regular updates from honest devices start to look similar, allowing malicious updates to blend more easily, Xia *et al*. [2023]. That is why robust protection systems are necessary to detect and prevent these hidden attacks, ensuring that the global model remains accurate, fair, and secure for everyone, Sun *et al*. [2024]. Hence, it is important to consider the behavioral patterns and data contributions of

each client in order to identify the relevance of the model's data to legitimize the client's nature, safeguarding the model against poisoned updates while preserving the accuracy and integrity of the FL system.

The state-of-the-art introduced four types of attack in which malicious clients alter their model updates to include poor-quality weights, leading to incorrect predictions and compromised system integrity. Specifically, label flipping pushes the model toward the wrong class and adds bias. Zero attacks pull weights back and hide significant changes, thereby shrinking the signal. The label shift changes the order of the channels, and thus regular checks cannot align the features. Random noise adds extra variation, making the data appear messy and rendering standard tests less reliable. Varying the attack mixture across rounds further degrades temporal anomaly detection; together, these coordinated effects suppress multiple anomaly metrics simultaneously, making malicious updates difficult to detect. Hence, it is important to design a resilient method that considers all four attacks occurring simultaneously, thereby representing a realistic and more challenging scenario Ariffin *et al*. [2025].

A method to mitigate the impact of these malicious clients is to use similarity algorithms, such as k-means, cosine cluster index, or Centroid-Based Kernel Alignment, to group clients. These algorithms create clusters to identify outdated models or malicious clients capable of compromising the global model, thereby protecting the global model against poisoning attacks Xia *et al*. [2023]. Clustering methods employ a pairwise technique to compare each client model, significantly increasing the instances needed to group the clients into different subjects. In some cases, their actual cluster may not correspond to the group to which they should belong, e.g., an honest client who possesses poor or corrupted data may cause clustering methods to mistakenly group honest clients with malicious clients. In this sense, it is important to refine the clustering method to handle various poisoning attacks while reducing computational costs to mitigate incorrect method choices. For these attacks, FL systems require a resilient method to contain the flow of malicious clients during aggregation, which is essential to mitigate the issues of global model generalization and convergence, making them more unstable and less accurate. Hence, the time and processing demands required to maintain the accuracy and integrity of the FL system against poisoned updates, while achieving faster and lower processing updates, remain open issues.

This article introduces MONZA, a scoring method for detecting malicious clients in FL environments, which incorporates a method that classifies clients by model. MONZA computes client scores using two complementary factors: model gradients and similarity. The method normalizes and integrates these metrics to determine a client's rank, which identifies the clients that are relevant for the aggregation. A quarantine method also helps increase similarity by comparing a consistently smaller number of clients. This method avoids relying solely on similarity scores, which may exclude clients who provide valuable updates. MONZA incorporates gradient evaluation to better assess each model's contribution.

MONZA employs cosine similarity to score and rank client models, assigning lower values to clients that are more deviant from the norm. When similarity scores become too close to distinguish between clients, particularly under label-flipping attacks, MONZA uses the L2 norm as an alternative metric, as clients exhibit remarkable similarity. Especially when we reduce the number of customers per round due to the decrease in customers resulting from the quarantine, we become increasingly cohesive in terms of similarity. A more powerful metric is necessary because the cosine metric is not a perfect match for pairwise comparison. This dual approach ensures robust evaluation across diverse attack types.

Our experimental evaluation shows that MONZA outperforms zPROBE (i.e., existing resilient methods) in terms of accuracy in a scenario with 30% malicious clients. For instance, MONZA achieves an accuracy of 54.5%, while zPROBE reached an accuracy of 50%. It is important to mention that default FL (i.e., FedAvg without malicious clients) reaches an accuracy of 55%, which means that MONZA performs very close to the benchmark method. In addition, MONZA incurs a minor cost in terms of MFLOP/s compared to existing resilient method, achieving 83 MFLOP/s, while our RALL achieves 297 MFLOP/s per round in a scenario with 30% malicious clients.

This article extends the previous work by Veiga *et al*. [2025], where its main research contributions can be summarized as follows:

- We rely on the cosine similarity score to identify malicious clients based on their individual models. When clients are highly similar, we also apply the L2 norm of gradients to assess divergence among clients more accurately, and then place those clients with an L2 norm in quarentine.
- We also reduce computational complexity by flattening the model parameters instead of processing each parameter as a separate matrix. In addition, we improve the comparison method by evaluating the divergence using the standard deviation of the similarity score matrix.
- The evaluation results demonstrate the efficacy of our proposed scheme, which significantly reduces the computational complexity per-round and decreases the time required for each round.

The remainder of this article is organized as follows. Section 2 presents an overview of the works that explore the similarity and response of defense to poisoning attacks. Section 3 describes our methodology to protect our aggregation and the global model resilience method. Section 4 explores the simulation model and the results obtained related to our and other methods. Finally, Section 5 concludes the article and provides directions for future work.

2 Related Works

Ghodsi *et al*. [2023] introduced a method that incorporates statistical limits in zero-knowledge proofs to identify and eliminate malicious updates while protecting user privacy, called zPROBE. This method ensures Byzantine resilience and secure FL, but it fails to detect malicious clients. zPROBE also does not consider computational issues arising from the numerous comparisons during the pairwise parameter evaluations. zPROBE aids in understanding how the cluster func-

tionality detects client model similarity and categorizes each model to facilitate improvements through others' evaluations of the client model. However, zPROBE does not consider layer attacks that significantly impact the comparison of the methods, as the method cannot be utilized when the attack is more sophisticated, since it introduces the same values in different layers.

Yan *et al*. [2023] proposed a defense method that uses an FL gradient norm vector (FGNV) to detect subtle but significant discrepancies in updates of the DNN model, called DeFL. This fine-grained method enables DeFL to identify malicious clients and specific compromised local participants (CLPs) that are excluded from the aggregation phase. However, DeFL does not take into account the parameters of the client's model. In this sense, this narrow focus can lead to an incomplete assessment of client trustworthiness, as the method may lack the necessary contextual information to accurately distinguish between benign and malicious clients. In other words, they cannot guaranty a generalization of the method to any attack strategy.

Rezaei *et al*. [2025] introduced a framework that integrates the concept of FL systems with Large Language Models for the real-time detection of anomalies in 5G networks, called FedLLMGuard. In contrast to traditional ML methods or time series training, the FedLLMGuard employs excessive computational effort when using Tiny-BERT for detection. Although it is lighter than the original BERT model, there are still several difficulties in training and synthesizing realistic attack traffic in the network. Although a model that offers excellent potential and delivers good results is used, the computational effort and response time can be significantly higher than those of simpler methods that can achieve similar results with fewer issues. However, FedLLMGuard demonstrated the need to protect the FL models in a generalized form, which means a more complicated scenario where we do not have a unique method to detect malicious clients, but rather various types.

Yazdinejad *et al*. [2024] presented a privacy-preserving federated learning (PPFL) model to protect against model poisoning attacks while maintaining data privacy and ensuring efficient training. The system employs additive homomorphic encryption (AHE) using the Paillier cryptosystem to ensure that gradients remain secure throughout the entire FL process, allowing safe calculations on encrypted data. A key feature is the use of an internal auditor that applies Gaussian Mixture Models (GMM) and Mahalanobis distance (MD) to identify and remove malicious gradients in both independent and identically distributed (IID) and non-IID data distributions.

Ma *et al*. [2022b] introduced ShieldFL, a privacy-preserving defense strategy designed to thwart encrypted model poisoning attacks within FL systems. This framework addresses the critical vulnerability where Byzantine adversaries submit malicious local gradients in encrypted form, bypassing traditional defenses that require plaintext data. A key challenge in FL is handling heterogeneous data; benign gradients trained on non-IID data can appear divergent, making them difficult to distinguish from malicious ones. ShieldFL overcomes this with a Byzantine-tolerant aggregation method, which uses secure cosine similarity to identify threats among encrypted gradients. This method ensures robust protection for IID and non-IID data distributions, effectively mitigating attacks without compromising user privacy. However, this method also does not consider a situation of multiple attacks for the security of the global model; in that situation, it is usual to determine the malicious update in a more complex solution.

Cao *et al*. [2022] introduced FLCert, an ensemble framework for FL designed to offer provable security against poisoning attacks. Unlike traditional methods, FLCert divides clients into groups and trains a separate global model for each, using a majority vote among the models to classify test inputs. FLCert guaranties that the model output is unaffected by a bounded number of malicious clients, regardless of the type of poisoning attack they attempt. FLCert includes two variants: FLCert-P, which probabilistically samples clients, and FLCert-D, which assigns clients to deterministic groups. The system is provably secure against poisoning attacks, providing a certified security level for each input. However, this method does not consider the intensity of various types of attacks, as it was only described on a specific type of attack; that is, label prediction. This means that if the attack does not lead to a label shift during training, this step does not apply effectively.

Zhao *et al*. [2025] introduced a defense system to protect FL from Byzantine poisoning attacks. This work addressed the issue of malicious clients by transmitting harmful updates that can disrupt the global model. Unlike traditional methods that assume a certain number of malicious clients, FedMP seeks unusual patterns in the direction and size of model updates. It employs a method to scale down significant malicious updates and group similar updates together, filtering out those that seem suspicious. Finally, FedMP uses pure ingredient-based aggregation, extracting key features from filtered updates to strengthen the global model. However, FedMP does not consider the computational cost of creating the clusters in this manner, particularly as the number of clients increases, rendering this method non-scalable due to the clustering method's need for a faster and more resilient method.

Table 1 provides a comprehensive overview of the key attributes of the reviewed studies on malicious client detection, Computational effort, and exclusion methods. Based on the state-of-the-art analysis, adopting a robust FL aggregation method is imperative to address the challenges posed by multiple malicious clients in FL environments. Therefore, the efficiency of detection through a ranking score method contributes to a lower computational effort, which also reduces energy costs. To achieve this objective, insights gleaned from other FL studies must be leveraged to accelerate the identification of the most suitable malicious client detection methods. Consequently, it is important to design a method that relies on cosine similarity analysis to compute the scores of clients' models, thereby establishing a robust methodology for detecting and eliminating malicious clients. To the best of our knowledge, only MONZA considers every critical characteristic previously mentioned that is not provided by the existing methods.

Table 1. Related Works

Work	Detection Method	Computational Effort	Exclusion Method
Xia *et al*. [2023]	Gradient magnitude analysis with threshold-based filtering	Low	Gradient norm threshold
Ghodsi *et al*. [2023]	Model similarity using intermediate activations	Medium	Low cosine similarity cluster
Yan *et al*. [2023]	Differential privacy with anomaly detection in update patterns	High	Statistical outlier detection
Rezaei *et al*. [2025]	Behavioral analysis of LLM-specific attack patterns	High	Pattern-based filtering
Yazdinejad *et al*. [2024]	Statistical hypothesis testing on model updates	Medium	Hypothesis test rejection
Ma *et al*. [2022b]	Similarity-based verification using auxiliary datasets	Medium	Low similarity threshold
Cao *et al*. [2022]	Certified robustness through geometric analysis	Medium	Certification failure
Zhao *et al*. [2025]	Model parameter verification via consensus checking	Low	Consensus violation
MONZA (Ours)	Dual-metric scoring combining cosine similarity and L2 norm	Low	Low ranking score threshold

3 MONZA

This section introduces MONZA, our method designed to enhance the resilience of FL against malicious clients using a dual score evaluation method. The MONZA operations begin by calculating a client score using the client's model parameters, establishing a comparative method for clustering, and then normalizing it to obtain a client score. As soon as the clients' similarity becomes abnormal, we use the L2 norm calculated from the gradients of the clients' models at the edge, achieving the most robust score behavior among the clients. For reincident clients, we use our quarantine method, which protects the aggregation policy from clients who repeatedly attempt to attack it. In this sense, MONZA mitigates the risks posed by aggregate malicious clients and non-valuable clients, which can decrease global model parameters and lead to incorrect evaluations. That operation not only improves the accuracy of malicious client detection, but also uses an optimized method by flattening parameters. Hence, MONZA reduces the computational time required to evaluate clients per round. In this section, we provide an overview of the scenario and MONZA operations.

3.1 Scenario overview

We consider a scenario with N devices, represented as $C_i \in \{C_1, \ldots, C_N\}$. The FL system begins each communication round by selecting K client devices, called *Clients*, which receive the global model and train it on their local dataset D_i. The selection process aims to choose clients with useful data samples while avoiding those whose information no longer contributes to improve the model, thereby reducing unnecessary computations. Each dataset D_i contains a set of features $x_{k,i}$, for $k \in \{1, \ldots, \|D_i\|\}$, where each feature has a corresponding label $y_{k,i}$. Using their local dataset, the chosen clients update the model Mn. Table 2 summarizes the main symbols used to describe the MONZA method.

Table 2. List of Symbols

Symbol/Acronym	Description
FL	Federated Learning
MONZA	Our resilient method
FedAVG	Federated Averaging
N	Total number of devices/clients
K	Number of clients selected per round
C_i	A single client device
D, D_i	Dataset (Global and Local, respectively)
x, $x_{k,i}$	A data feature/input
y, $y_{k,i}$	A label/output
$p_i(x)$	Probability distribution of data point x for client C_i
θ	Model weights/parameters
θ_{global}	Global Model parameters
$\mathbf{A}$, $\mathbf{B}$	Vectors for similarity calculation
$\mathbf{Q}$	Quarentine dictionary for malicious clients
β	Concentration parameter for Dirichlet distribution
α	Mixing strength parameter in attacks
FLOPs	Floating Point Operations
Non-IID	Non-Independent and Identically Distributed
n_i, n	Number of samples (local and total)
$\mathbf{g}$	Gradient vector
ε, ε_l	Noise vector in random attack
σ_l	Standard deviation in random attack
s, t	Source and target classes in label-flipping attack
$P_{\text{in}}^{(l)}$, $P_{\text{out}}^{(l)}$	Permutation matrices in shuffled-layer attack
q	Fraction of malicious clients in zero attack analysis
$\rho^{(t)}$	Alignment metric at round t
$L2_{\text{grad}}$	Client L2 gradient
S	Similarity Matrix

In FL, probability distributions describe how data spread across clients. Each client C_i has a local data set D_i, and the complete data available in the federation is given by the union of all local datasets as in Eq. 1.

$$D = \bigcup_{i=1}^{K} D_i \tag{1}$$

For each client C_i, we define a probability distribution p_i over its dataset D_i as shown in Eq. 2.

$$p_i(x) = \frac{\text{Occurrences of } x \text{ in } D_i}{\text{Total elements in } D_i} \tag{2}$$

This distribution $p_i(x)$ expresses the probability of observing a data point x within the local client dataset C_i. The FL training process builds a global model by combining client information. The aggregated model parameters θ are updated in each round by averaging the contributions of individual clients as shown in Eq. 3.

$$\theta_{global} = \frac{1}{K} \sum_{i=1}^{K} \theta \tag{3}$$

After training, the clients return their model weight updates θ to the edge server. These updates may include either the learned parameters or the computed gradients. In this way, every device contributes its unique data, which helps to build a more complete and reliable global model. The server then applies the FedAVG aggregation strategy, which combines the updates into a single global model by computing the mean of all local models Zhang *et al*. [2023]. This aggregation step on the server is essential for merging client contributions and producing an improved global model that reflects the combined progress of all participants. The improved global model is then redistributed to the selected clients for the next round. Here, θ_{global} denotes the global parameters, while θ corresponds to the parameters updated by the client C_i. These local probability distributions $p_i(x)$ highlight the diversity of data among clients. Measurement of this diversity is essential, and entropy becomes a key analytical tool.

Figure 1 shows the proposed FL scenario, emphasizing the malicious attack of the client. In this way, the edge server relies on MONZA to compute the cosine similarity score based on the clients' weights, checking whether they meet the requirements or converge similarly to the global model. The system ranks all clients by comparing their weights with the global model, giving higher scores to those whose data are more similar to the training data. If a client behaves maliciously, the system assigns a very low score and moves it to a separate list of excluded clients. Finally, after excluding malicious clients and allocating them using a quarantine method, we remove them from the aggregation in rounds of 2^n, where n represents the number of times they were removed from the aggregation. The edge server aggregates the remaining clients to build a new global model. Then, it sends this model back to all malicious and regular clients Song *et al*. [2022].

3.2 Multiple Attack Effort Comparative

As mentioned, each client C_i trains local models over its private data D_i and shares updates with an edge server for aggregation. However, malicious clients could manipulate these updates to poison the model, leading to incorrect predictions and compromised system integrity. In this paper, we consider four types of attacks: zero attack, layer shuffling, random values, and label flipping.

The Zero-Parameters attack allows a malicious client to sabotage the model's training by either sending zero weights ($\theta_k = \mathbf{0}$) or a weight update that cancels the current model ($\Delta\theta_k = -\theta^{(t)}$), as shown in Eq. 4 and described by Guo [2023]. Both methods shrink the global model's parameters toward zero, effectively reversing learned signals that lead to inadequate fitting or reversal. This deliberate reduction in magnitude and alignment ($\rho^{(t)} \downarrow$) prevents the model from learning meaningful features, halting or even reversing progress.

$$\Delta\bar{\theta}^{(t)} \approx (1-q)\bar{\mathbf{g}}^{(t)} - q\,\theta^{(t)}. \tag{4}$$

The Shuffled-Layer Parameters attack uses feature misalignment to disrupt model averaging, as described by Gu *et al*. [2023]. For each layer l with weights θ_l and bias $\mathbf{b}_l$, the attacker's model applies permutations $P_{\text{in}}^{(l)}$ and $P_{\text{out}}^{(l)}$, as shown in Eq. 5. This misalignment of the parameters during aggregation erodes the structure of the model, reduces $\rho^{(t)}$, and destabilizes downstream statistics such as BatchNorm.

$$\theta_l' = P_{\text{out}}^{(l)}\,\theta_l\,\left(P_{\text{in}}^{(l)}\right)^{\top}, \qquad \mathbf{b}_l' = P_{\text{out}}^{(l)}\,\mathbf{b}_l. \tag{5}$$

In a random attack, the malicious client replaces its parameters (or updates them) with noise to drown out the honest signal as described by Ghodsi *et al*. [2023]. Concretely, for each layer l, it reports similarly for biases, or it sends a pure-noise update as shown in eq. 6 $\Delta\theta' = \boldsymbol{\varepsilon}$. By tuning α (and σ_l) to match the norm of honest updates, the attacker preserves the outward "scale" while injecting high variance and a near-zero mean, which lowers the signal-to-noise ratio of the server aggregate. As a result, the global step aligns less with the proper descent direction, the training slows down, and with momentum or a greater η, the model can oscillate or diverge. Variants include Rademacher noise ($\varepsilon_i \in \{-1, +1\}$) or random sign flips of coordinates to maximize cancelation without triggering simple norm-based defenses.

$$\theta_l' = (1-\alpha)\theta_l + \alpha\,\boldsymbol{\varepsilon}_l, \qquad \boldsymbol{\varepsilon}_l \sim \mathcal{N}(\mathbf{0}, \sigma_l^2\mathbf{I}), \tag{6}$$

In the label flip attack, the malicious client remaps a source class s to a target class t, as described byZang and Li [2024]. In this sense, this attack introduces bias in the gradient, which leads to an incorrect decision boundary. For each (x_i, y_i), the poisoned labels can be expressed as shown in Eq. 7.

$$y_i' = \begin{cases} t, & y_i = s, \\ y_i, & \text{otherwise}, \end{cases} \tag{7}$$

The malicious client trains on the poisoned dataset $\mathcal{X}_k' = \{(x_i, y_i')\}$ before submitting its parameters or updating. This training increases the logarithmic scores for the target class t in examples whose true label is s, causing the malicious update $\mathbf{a}_k$ to become negatively aligned with the honest gradient $\mathbf{g}_k$ in these samples, i.e., $\mathbb{E}[\langle \mathbf{a}_k, \mathbf{g}_k \rangle] < 0$.

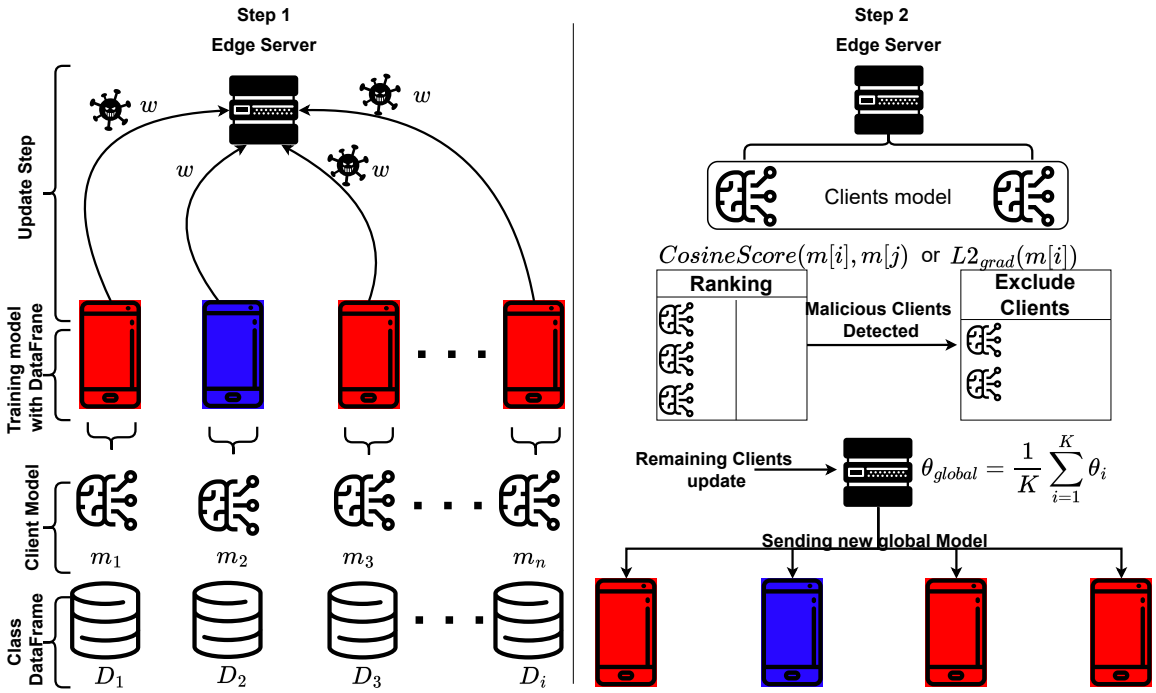

Figure 1. Scenario overview for resilient global model attacks

Eq. 8 optionally mixes honest and poisoned training with $\alpha \in [0, 1]$. The attacker maintains the update scale while injecting systematic bias into the client's model to decrease the global model update. That approach requires no rescaling and produces updates that closely resemble legitimate ones. After aggregation, the global update aligns poorly with the proper descent direction, causing the decision boundary to shift toward class t and potentially slowing or destabilizing training.

$$\Delta\theta'_k = (1-\alpha)\,\Delta\theta_{k,\text{clean}} + \alpha\,\Delta\theta_{k,\text{poison}}, \tag{8}$$

We consider a scenario in which all four attacks occur simultaneously, representing a realistic and more challenging situation. Label flipping pushes the model toward the wrong class, thus introducing bias. Zero attacks pull weights back and hide significant changes (shrink the signal). The label shift changes the order of the channels; thus, regular checks cannot align features. Random noise adds extra variation, making the data look messy and making standard tests less reliable. Varying the attack mixture over rounds further degrades the detection of temporal anomalies. Together, these coordinated effects suppress multiple anomaly metrics simultaneously, making malicious updates difficult to detect Ariffin *et al*. [2025].

3.3 Similarity Score

MONZA helps to select a subset of reliable clients K, but for our method, we focus solely on building a resilient system to protect the global model. In this context, we do not consider the client data metrics to select clients. Instead, we rely on all available clients to contribute their updates, employing a standard method such as the FedAvg algorithm to aggregate these updates.

In this manner, MONZA relies on cosine similarity to comprehend the similarity among clients. These clients also exhibit a small difference, suggesting that they share a similar model by training on analogous types of data that include the same dataset, resulting in the same data distribution. Similar data create similar models for comprehensive model convergence. Given a dataset distribution, a honest client can experience a decrease in model aggregation accuracy. At the same time, the development of attacks can have a model-convergence effect that is very difficult to distinguish from a malicious client with poor data. The extraction of features that have not necessarily been trained on a poor dataset indicates incorrect data filtering. This is because not all cases involving label-flipping lead to a complete attack. In the model, the more accurate archives correspond to the classification label. Therefore, it can contribute in some cases if the model has not already reached its convergence.

MONZA considers a quarantine method with the purpose of not selecting recurrent clients identified as malicious, as detailed in Algorithm 1. Once the system detects this client, it must use the dictionary to record how many times it has categorized the client as malicious in a round. If a new detection occurs, it must increase the number of rounds, and the client skips the training step to be selected.

The initial step involves setting up the $\mathbf{Q}$ dictionary for all N clients, which is a necessary method to map clients in our FL system. The dictionary helps to elaborate on the improvement in server control, recording both the detection counter ('count') and the duration of quarentine ('rounds'). In each subsequent round, the server first updates the status of all clients by decreasing the value $\mathbf{Q}[k]['\text{rounds}']$. This ensures that the count never falls below zero, eventually allowing quarentined clients to regain eligibility.

The crucial step is the application of the penalty. For a

newly detected malicious client k, the system increments the value $\mathbf{Q}[k]['\text{count}']$ (denoted as n). This metric elaborates on how our questionnaire evaluation excludes clients that do not improve our scenario and could decrease the accuracy. The system explicitly excludes quarantined clients from the set $S_{available}$ (eligible clients), preventing their selection and subsequent pollution of the global model. The duration of quarentine, $\mathbf{Q}[k]['\text{rounds}']$, then increases exponentially using the expression $\mathbf{Q}[k]['\text{rounds}'] + 2^n$. This exponential growth ensures that the penalty for recidivism is severe, discouraging malicious behavior in the scenario of attacks, which is a necessary method to contain the number of attacks. That method forces the client to skip the ClientUpdate training step for an increasing number of rounds before being considered for selection by RandomSample again. The system explicitly excludes quarantined clients from the set $S_{available}$ (eligible clients), preventing their selection and subsequent pollution of the global model.

Algorithm 1: Quarentine method

Data: Global Model θ_{t-1}, Selection Size K, Quarentine Dict. $\mathbf{Q}$
Result: New Global Model θ_t, Updated $\mathbf{Q}$

Server executes at round t**:**
Update Client Status: for *each client* k *in all* N *clients* **do**
  if $\mathbf{Q}[k]['\textit{rounds}'] > 0$ **then**
    $\mathbf{Q}[k]['\text{rounds}'] \leftarrow \mathbf{Q}[k]['\text{rounds}'] - 1$
  end
end
$S_{available} \leftarrow \{k \mid \mathbf{Q}[k]['\text{rounds}'] = 0\}$
$S_t \leftarrow \text{RandomSample}(S_{available}, K)$
for *each client* $k \in S_t$ *in parallel* **do**
  $\theta_k^t \leftarrow \text{ClientUpdate}(k, \theta_{t-1})$
end
$M_t \leftarrow \text{MaliciousDetection}(\{\theta_k^t \mid k \in S_t\})$
$\theta_t \leftarrow \texttt{AggregateFedAvg}(\{\theta_k^t \mid k \in S_t \setminus M_t\})$
Apply Penalty: for *each client* $k \in M_t$ **do**
  $\mathbf{Q}[k]['\text{count}'] \leftarrow \mathbf{Q}[k]['\text{count}'] + 1$
  $n \leftarrow \mathbf{Q}[k]['\text{count}']$ $\mathbf{Q}[k]['\text{rounds}'] \leftarrow 2^n$
end

MONZA starts after uploading the client data, as detailed in Algorithm 2 (line 1). Each client flattens its own model parameters into one-dimensional arrays (line 2), which simplifies the structure of the parameters and makes it easier to compare the overall model among clients. In this sense, MONZA computes the pairwise cosine Similarity between the model parameters submitted by all participating clients (line 3). Cosine similarity is a crucial geometric metric for measuring the angle between two vectors, e.g., A and B, in a vector space. In our context, A and B represent the flattened parameter vectors of two clients, and Eq. 9 denotes the fundamental relationship defining the similarity based on the angle Ω. In this case, S initially represents the coefficient where the cosine similarity value is located. Therefore, S is no longer a matrix but rather a similarity value normalized within the range from 0 to 1, taking into account the mentioned off-diagonal values. However, with these values already normalized by the function, σ means the average value.

$$\text{Similarity}(\mathbf{A}, \mathbf{B}) = \cos(\Omega) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{9}$$

Algorithm 2: MONZA (concise): Cosine Similarity and low-variance branch

Input: *uploaded_models*, *ids*, *index_malicious*
Output: S and *scores or grad*
`// 1) Similarity matrix`
$N \leftarrow |\textit{uploaded\_models}|$; $\varepsilon \leftarrow 10^{-10}$;
$\mathbf{F} \leftarrow \text{stack}(\text{Flatten}(\textit{uploaded\_models}[i]))_{i=1}^{N}$;
$S \rightarrow CosineSimilarity(F)$;
$\sigma \leftarrow \text{std}(S)$;
`// 2) Low-variance branch: return per-client L2 gradient`
if $\sigma < 0.001$ **then**
  for $i \leftarrow 1$ **to** N **do**
    $\textit{grad}[\textit{ids}[i]] \leftarrow$ L2Grad($\textit{uploaded\_models}[i], \lambda$=0.01)
  end
  return $S, \textit{grad}$;
end
`// 3) Normal case:`
for $i \leftarrow 1$ **to** N **do**
  $\textit{scores}[\textit{ids}[i]] \leftarrow \frac{1}{N-1} \sum_{j \neq i} S_{ij}$
end
$\textit{scores} \leftarrow \textit{sort}(\textit{scores}[\textit{ids}[i]])$
`// 4) threshold` $\mu - \sigma$`; reverse-order removal`
$\mu \leftarrow \text{mean}(\{\textit{scores}[\textit{ids}[i]]\})$; $\sigma \leftarrow \text{std}(\cdot)$;
$\tau \leftarrow \mu - \sigma$;
for *idx* $\leftarrow N-1$ 0 **do**
  $id \leftarrow \textit{ids}[\textit{idx}]$; $u \leftarrow \textit{scores}[id]$;
  if $u < \tau$ **then**
    set_client_quarentine(id); *uploaded_models*,
  end
end
return $S, \textit{scores}$;

The similarity $\in [0, 1]$ is determined by the ratio of the dot product to the product of the vectors' magnitudes, as shown in Eq. 10. The primary benefit of this metric is its independence from magnitude. A higher similarity value (closer to 1) indicates that the two clients' models are moving in consistent directions within the weight space, suggesting honest behavior. This is related to our previous indication that a more similar client has a point improvement in this pairwise comparison to continue with our aggregation. However, a low value indicates significant divergence, a characteristic often used to flag potentially malicious or data-poisoned models.

$$\text{Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{10}$$

We consider a client to be malicious or lacking a suitable model when the cosine similarity score between two clients

falls below a predefined threshold (line 4). In this way, we exclude these clients from further comparisons or aggregation, and thus we remove clients whose models are too different, ensuring that only similar models contribute to the global model.

The algorithm computes, for each client, the standard deviation of its cosine scores against all other clients. These cosine score profiles provide a cohort-level view of alignment. However, certain biases, such as label flipping, may not be evident in raw cosine scores alone if a client's cosine scores and standard deviation fall below a small threshold (for example $\epsilon = 0.001$) (line 5). Cosine similarity difference minus one standard deviation quantifies similarity levels. Very low values of this measure show highly aligned gradients, which help detect more precise and malicious attacks sorting their score values. The model appears to be overly stable and potentially biased (that is, it has learned less variation than its peers).

We established the threshold based on empirical experiments and the heuristic $\mu = \mu - \sigma$ (an approximation of the lower bound of the confidence interval). In this sense, values $\leq\sim 10^{-3}$ indicate extreme alignment of the gradients (in the $[0, 1]$ range), justifying exclusion due to redundancy or suspicion of attack. The combined use of cosine similarity and divergence analysis allows discarding both highly similar updates (low informativeness or malicious) and excessively divergent ones (noise or poor quality), thereby promoting a more robust and informationally diverse aggregation.

We apply L2 normalization (L2Grad) to the parameter vector for such clients to remove magnitude effects and emphasize direction (lines 6-10). We then recompute the cosine scores between the $L2Grad$ parameters of the flagged client and those of all the other clients. We exclude the client from aggregating into the θ_{global} if the recomputed cosine scores exceed the acceptance threshold (τ), as shown in Eq. 11 for each client, where N_{param} is the number of parameters. In summary, we rely on cosine scores to compare client models and exclude those that are highly dissimilar. When cosine scores are too close to the standard deviation indicating potential bias (lines 11-13), MONZA applies $L2Grad$ to score the clients sorting their scores (line 14). We exclude clients who still show a low standard deviation of cosine scores after $L2Grad$, ensuring that only relevant, less biased contributors influence θ_{global} (line 15).

$$L2_{\text{Grad}}(\theta) = \frac{1}{N_{param}} \sum_{i=1}^{N_{param}} \|\theta\|_i^2 \tag{11}$$

Finally, if the recalculated cosine similarity (after L2 normalization) remains below the threshold, the client is excluded from the aggregation process in reverse order (lines 16-21), because the algorithm sorts by ID and the ranking needs to use the clients' ranking value. This issue ensures that only clients with sufficiently similar models contribute to the global model. MONZA considers cosine similarity to compare client models and excludes those that are too dissimilar from each other (lines 16-21). Additionally, suppose a client's model is too stable (indicating bias). In that case, it undergoes L2 normalization and recalculates the similarity, which is important to determine whether it is truly a client with reliable bias. Clients that continue to show low similarity after excluding normalization. Hence, MONZA ensures that only relevant and unbiased clients participate in the aggregation and contribute to a more accurate global model (line 22).

4 Evaluation

This section presents the simulation setup used to evaluate MONZA's comparison to existing methods, as well as state-of-the-art and previous works. We describe here the simulated FL scenario, including the underlying framework, the characteristics of the database, and the simulation parameters. Following this, we present the results obtained to assess the effectiveness of MONZA relative to the baseline methods. Our simulation code can be find on the github[1].

4.1 Methodology

We conducted an extensive simulation study using the PFLLib[2], a framework introduced by Zhang *et al*. [2023]. We executed the PFLLib framework on a server with the following specifications: Intel Core i9-13900K (32 cores), 128 GB RAM, and dual NVIDIA RTX 4090 GPUs, running on an Ubuntu Server operating system. We used widely public datasets for training and testing model validations (i.e., CIFAR-10 and CIFAR-100), which features color images of animals and objects, and dataset MNIST which features are 10 classes of numbers Zhang *et al*. [2023]. We use a batch size of 10, a learning rate of 0.005, and 1 epoch of training as the main hyperparameters. Results show the values with a confidence interval of 95%. Table 3 summarizes the simulation setup and parameters.

Table 3. Simulation Parameters

Description	Values
Number of Clients	100 Clients
Malicious clients	30% of all clients
Datasets used	CIFAR-10, CIFAR-100 and MNIST
Number of clusters	2 clusters
Selected Clients	100% of clients
Local epochs	1 epoch
bach size	10 bachs
Optimizer	Stochastic Gradient Descent
Learning rate	0.005
Learning rate scheduler	Exponential Learning Rate Decay
Number of Executions	10 Executions
Confidence interval	95%
Resilient methods	MONZA, zPROBE, RALL
Attacks used	all: Zero method, Shuffled-Layer, Random Attack and Label flipping attack

[1]MONZA available at `https://github.com/VeigarGit/PFLlibMonza`

[2]https://github.com/TsingZ0/PFLlib

The simulation models an FL system composed of 100 clients, where we consider 30% of malicious clients with a diverse set of attacks. Malicious clients are created by randomly assigning each malicious client to one of the four types of attacks (i.e., Zero-Parameters, Shuffled-Layer Parameters, random, and label-flipping) with equal probability (1/4). In this sense, each client's impact during an attack depends on the attack type in that round. During the simulation, a malicious client can choose one of those attacks in each round, making it difficult to develop a pattern for detection because the client uses different attack complexities during the same simulation.

We employ a CNN model architecture with two convolutional layers using 5x5 filter sizes, followed by 2x2 max-pooling operations after each convolutional layer. We employ non-IID data throughout the experiments to ensure a realistic representation of data distribution challenges modeled using a Dirichlet distribution with a concentration parameter of $\beta = 0.2$. We use label distribution to characterize the local data distribution among clients by a proportion, as described in Ma *et al*. [2022a].

We consider the following evaluation configuration: i) the Default FL consists of random client selection without malicious clients, which serves as a benchmark result that does not suffer from any attacks; ii) we also assess the performance of Default FL in a scenario with malicious clients to compare its efficacy in selecting clients (referred to as without defense in our results); iii) attacked with defense: zPROBE, RALL, and MONZA. Specifically, zPROBE considers a cluster-based method that excludes clients from selected clusters. RALL, our previous work with a similar parameter method, differs in that it does not utilize L2 norm-based scoring or parameter flattening. Finally, MONZA is our method that, in addition to using a similarity method with two parameters to identify malicious clients, calculates a score using the cosine similarity and the L2 norm-based approach to remove unnecessary clients in that round.

We evaluate performance using key performance metrics to gage the effectiveness and efficiency of MONZA compared to state-of-the-art. Specifically, **Accuracy** measures how well a model classifies or predicts data correctly, and is computed as the proportion of correct predictions out of the total predictions made as shown in Eq. 12. Accuracy helps to determine how closely the performance of MONZA aligns with the proper labels.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{12}$$

Entropy loss, also known as Cross-Entropy Loss, is an important metric for measuring the performance of classification models, especially when they provide probabilistic outputs. Shows the uncertainty or disorder between the predicted class probabilities and the actual labels, as shown in Eq. 13. Specifically, y_i denotes the actual label (one hot encoded), p_i means the predicted probability for the actual class, and N denotes the number of classes. A lower entropy loss indicates a better model performance, since it implies that the predicted probabilities are closer to the actual labels. In this sense, we can evaluate how much our method improves the certainty of predictions while reducing uncertainty.

$$\text{Entropy Loss} = -\sum_{i=1}^{N} y_i \log(p_i) \tag{13}$$

False Rejection Rate (FRR) is a metric that proposes to study a proportion of verification transactions with truthful claims of identity that were incorrectly denied. FRR is calculated using False Negative (FN) divided by FN plus true positive (TP). This metric occurs when a security parameter/system fails to recognize an authorized user, , a false rejection occurs. The FRR is calculated as Eq. 14 used to calculates this metric of verification.

$$\text{FRR} = \frac{\text{FN}}{\text{FN+TP}} \tag{14}$$

False Positive Rate (FPR) measures the proportion of verification transactions with false identity claims that the system incorrectly accepts. In this sense, we compute FPR as the Number of False Positives (FP) divided by FP plus true negatives (TN), as shown in Eq. 15. This metric indicates when a security parameter/system fails to reject an unauthorized user, i.e., a false acceptance happens.

$$\text{FPR} = \frac{\text{FP}}{\text{FP+TN}} \tag{15}$$

In addition, execution time plays a critical role in evaluating the practicality and efficiency of the model. The relationship between execution time and accuracy is often a trade-off. Although improving accuracy may sometimes lead to longer execution times, optimizing for both performance and speed is important, especially in real-time or resource-constrained environments. To evaluate the trade-off between execution time and accuracy, we measure the total time it takes for the model to process the data and produce predictions. We also consider the average client training time (ctt) to account for all computational effort.

We compute the total FLOP/s for the client i by summing the FLOPs for each layer of the model. For a fully connected layer, the number of operations depends on the number of input and output units, as each input unit multiplies with each output unit. The results are summed and calculated for each client i according to Eq. 16, which considers the number of operations in each model layer. L represents the total number of layers in the client model. For each layer l, the input size (input size$_l$) refers to the number of input units in that layer, while the output size (output size$_l$) refers to the number of output units from that layer. The factor of 2 represents the multiplication and addition operations typically performed in the computation of each neuron, where each input unit is multiplied by the corresponding weight and added to the bias.

$$\text{FLOP/s}_i = \sum_{l=1}^{L} 2 \cdot \text{input size}_l \cdot \text{output size}_l \cdot \text{ctt} \tag{16}$$

4.2 Results

Figure 2 shows the accuracy over the simulations using default FL without attacks, as well as MONZA, zPROBE, and

default FL with receiving attacks from a mix of 30 clients per round. The method without defense demonstrates the most catastrophic scenario in which malicious clients aggregate in all rounds and receive the four attacks simultaneously. That indicates a notable decrease in model accuracy of over 12%. This is a very low-performance scenario compared to the default method, which involves a special line to achieve the evaluation behavior of all clients training in the scenario, resulting in more than 50% of evaluations in the final round. On the other hand, zPROBE requires the exclusion of an entire cluster, which can lead to incorrect conclusions in the aggregation, especially during label-flipping attacks. The incorrect detection made in zPROBE has some resilient problems with its accuracy, reaching 48% accuracy in the final rounds. In addition, MONZA addresses the problem of clustering by cosine, using its punctuation to allocate more similar clients. With a small number of clients exhibiting behavior more similar to the defaults, while excluding malicious clients, it achieves 44.5% accuracy.

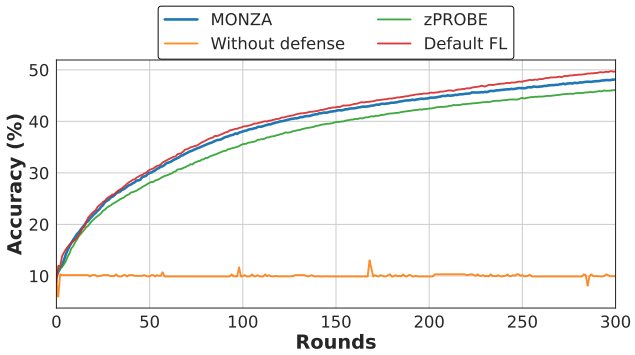

Figure 2. Accuracy Evaluation Results of Attack Impact.

Figure 3 shows the loss over the simulations using default FL without attacks, as well as MONZA, zPROBE, and default FL with receiving attacks from a mix of 30 clients per round. The method without defense behavior does not show an increase in its parameters relating to high divergence, as it experiences more than three losses in many cases. That occurs because the system undergoes a significant dislocation during the aggregation step. By aggregating malicious models, the loss value increases due to the similarities of the gradients. That differs when dealing with defensive methods, as it involves values of loss that are very similar to the Default FL, whose evaluation ranges from 2.4 to 200 rounds, with an evaluation of 1.5, which is very similar to MONZA and zPROBE. That is, clients with a similar gradient do not hinder the model's progression during training. Although the other three methods show similar behavior, they are also problematic due to their aggregation of unreliable clients: they have more loss value until MONZA is closer to the default method.

Figure 4 shows the accuracy values over time for the simulations using the MONZA method, zPROBE, and RALL. In this sense, we sum the time of each client train, the selection step, and the model update. The improvements in the results from the MONZA evaluation resulted in a decrease of almost 66% in execution time compared to RALL, which is the method of analyzing the parameters that adopts a more advanced approach. We have input on MONZA, which involves flattening all the parameters. This means that RALL

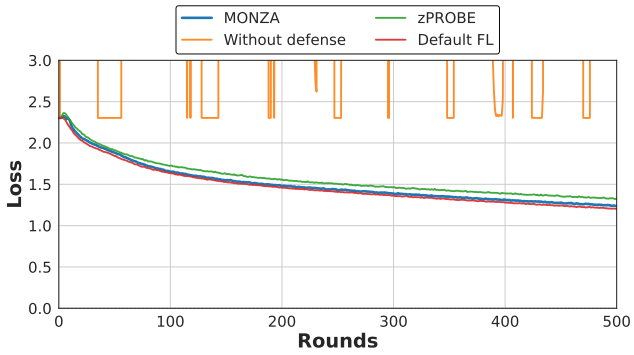

Figure 3. Loss Evaluation Results of Attack Variation Impact.

used a formulation in a matrix, which has a more complex cost for comparing their parameters. This is different from the one used on MONZA, where a simplified one-dimensional approach is employed. This decreases the complexity of the recursion in the method and facilitates the determination of the similarity matrix during the cosine extraction formulation. During the final simulation, MONZA achieves an accuracy improvement of almost 50% in approximately 27 minutes. Our evaluation shows that zPROBE, which uses that method with the flattening technique, achieves a time of 33 minutes per round. The figure also demonstrates the improvement achieved by RALL, which averages more than 55 minute in total time across the final simulation.

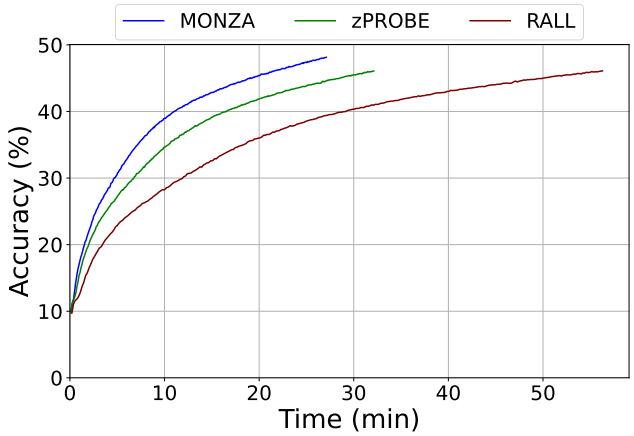

Figure 4. Accuracy by Time Evaluation Results of Attack Impact.

Figure 5 presents our two comparisons of MONZA and zPROBE in terms of FRR, identifying where clients were incorrectly identified due to misclassification issues. In the zPROBE method, we employ only the cosine similarity approach to compare clients' similarities, which poses a challenge. In some cases, the clients involved in shuffle or label flipping attacks are too similar to their original models. This results in zPROBE making mistakes when selecting clients, with an FRR exceeding 24% during all simulations. On the other hand, MONZA encounters some unstable peaks that are consistent with our method, considering that we are dealing with a variety of factors that require our method to increase by 2^n. That results in our FRR being approximately 22% in some exceptional rounds. However, in the first rounds, we do not have a consolidated model for clients; as a result, we made some mistakes in the initial rounds. The incorrect classification of honest clients must show that clients are not malicious, but do not have valuable parameters for our

aggregations because of their training data.

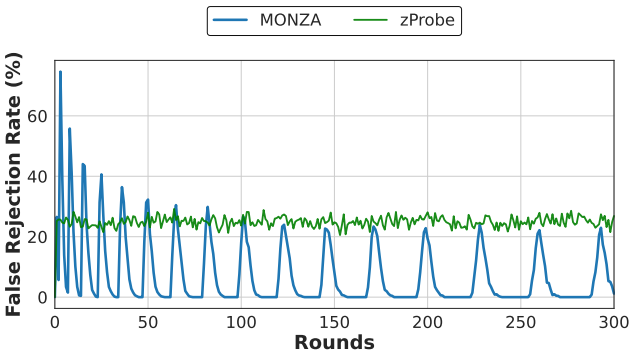

Figure 5. FRR Evaluation

As we use cosine similarity to L2 normalization to compare clients, we must understand where the L2 normalization focuses its values. That is, the system flags a client's model update as malicious when its L2 distance from the global average exceeds τ. This rule does not distinguish between malicious intent and legitimate statistical heterogeneity. Taking into account an honest client k whose data is non-IID, the best local model for this client, $\boldsymbol{\theta}_k^*$, is naturally different from the best global model, $\boldsymbol{\theta}^*$. The system checks how far the client's model is from the global one by measuring $D(k) = \|\boldsymbol{\theta}_k^t - \bar{\boldsymbol{\theta}}^t\|_2$. Using the triangle inequality, we can see that $D(k)$ is at least as large as $\|\boldsymbol{\theta}_k^* - \boldsymbol{\theta}_g^t\|_2$ minus a small margin of error. Eq. 17 describes this behavior, where $\mathbb{E}[\text{FRR}]$ depends on the probability that $|\boldsymbol{\theta}_k^* - \boldsymbol{\theta}_g^t|_2$ exceeds a threshold τ. As a result, the system places the honest client k in quarantine.

$$\mathbb{E}[\text{FRR}] \propto P(\|\boldsymbol{\theta}_k^* - \boldsymbol{\theta}_g^t\|_2 > \tau) \tag{17}$$

Eq. 17 shows that the known problem lacks real importance. We justify this by treating a structural problem of a non-IID system. When dealing with that type of distribution, we must consider the client's divergence, which means that a honest client does not contribute to the aggregation. Therefore, the model's gradient values indicate that aggregating that client and its convergence based on its data decreases the model's accuracy.

Figure 6 shows our evaluation of the methods RALL, zPROBE and MONZA in terms of computational effort, measured in MFLOP/s, which ensures that the method's behavior is suitable for running in our centralized environment. For these purposes, we used the round-time to calculate the behavior of each method when training the clients. That method showed a minimum decrease related to the zPROBE, but it was significantly different from our previous method, RALL. MONZA has a minor processing effort of 83 MFLOP/s due to the method used to optimize the client's parameters, which zPROBE can also observe. However, that method has a similar scope in extracting our MONZA, taking 118 MFLOP/s. At the same time, RALL does not have this optimization, which makes that method too slow compared to the others.

Table 4 presents the average performance metrics (accuracy, FPR, and FRR) for MONZA when evaluated against the default FL baseline on MNIST and CIFAR-100. MONZA achieves an accuracy of 97.11%, nearly identical to the

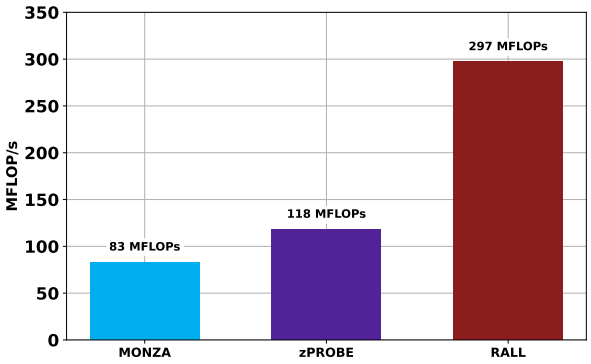

Figure 6. MFLOP/s Evaluation

97.25% of the default FL approach. Moreover, it maintains very low error rates, with an FPR of 0.7% and an FRR of 7.5%, confirming that MONZA reliably identifies malicious clients on simpler datasets where gradient similarities stay clearly distinguishable.

On the other hand, in the more complex dataset (i.e., CIFAR-100), MONZA requires a much stricter similarity threshold due to faster model convergence and smaller divergence among client updates. These results show that the values reach a threshold of 0.001, where clients' gradients become increasingly similar and cannot consistently adapt to the expected patterns. Consequently, although MONZA limits the accuracy degradation to less than 3% compared to the baseline, the rapid convergence of most clients' gradient norms leads to frequent misclassification of benign clients as malicious, resulting in average FPR and FRR values of 36.8% and 7.4%, respectively, comparable to those of other defense methods.

Table 4. Metrics of Detection Measurements on Different Datasets

Model	Method	Accuracy (%)	FPR (%)	FRR (%)
CIFAR-10	MONZA	47.58	3.6	7.68
	Without defense	10.11	-	-
	Default FL	50.07	-	-
CIFAR-100	MONZA	22.64	36.8	7.4
	Without defense	0.99	-	-
	Default FL	25.35	-	-
MNIST	MONZA	97.11	0.7	7.5
	Without defense	9.5	-	-
	Default FL	97.25	-	-

5 Conclusion

This article introduced MONZA, an FL system that detects and stops malicious clients. MONZA addresses the issue of malicious clients sending harmful updates that corrupt the shared global model. It gives each client a score based on the similarity of their model to others. Clients with a very low score are perceived as malicious and are temporarily blocked from participating, helping protect the model.

Our tests showed that MONZA works better than current methods such as zPROBE and RALL. In a problematic situation where 30 out of 100 clients attacked the FL system, MONZA maintained a model accuracy of 54.5%, higher than the others. It was also much faster, reducing the execution time by almost 66% compared to RALL. Finally, MONZA

required the least computer power, using only 83 MFLOP/s, confirming that it is a very efficient solution.

Therefore, MONZA is a robust, lightweight, and scalable solution to protect systems against subsequent poisoning attacks. Our method protects the integrity and convergence of the model while minimizing resource consumption, which improves the novel method. Future work focuses on adapting the MONZA method to defend against a broader range of adaptive attacks, as we show that the label flipping is an attack making direct on clients' data which does an attack more precisely, but also poisoning the classification of the clients and a decrease in the aggregation and exploring its application within personalized federated learning scenarios to enhance its robustness and versatility further.

Acknowledgments

This research was partially funded by the CNPq grant #405940/2022-0, CAPES grant #88887.954253/2024-00 and PRR — Plano de Recuperação e Resiliência and by the NextGenerationEU funds at Universidade de Aveiro, through the scope of the Agenda for Business Innovation ''ATT — Agenda Mobilizadora Acelerar e Transformar o Turismo'' (Project no. 47 with the application C645192610-00000060). This study was also funded, in part, by the São Paulo Research Foundation (FAPESP), Brazil, under Process Number <2023/00673-7>. This project was supported by the Brazilian Ministry of Science, Technology and Innovations, with resources from Law No. 8,248, of October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex, and published under Arquitetura Cognitiva (Phase 3), DOU 01245.003479/2024-10.

Authors' Contributions

R.V. and R.M. conceived and planned this study and experiments. R.V. and R.M. carried out the experiments, the simulations, and computed the results. L.B., S.S, D.R., E.C. contributed to the interpretation of the results. R.V. took the lead in writing the manuscript with final review by L.B., S.S., D.R., and E.C. All authors provided critical feedback and helped shape the research, analysis and manuscript.

References

Ariffin, A., Zaki, F., Hanif, H., and Anuar, N. B. (2025). Adversarial attack and defence of federated learning-based network traffic classification in edge computing environment. *Computer Networks*, page 111739. DOI: 10.1016/j.comnet.2025.111739.

Barros, A., Veiga, R., Morais, R., Rosário, D., and Cerqueira, E. (2024). Mesfla: Model efficiency through selective federated learning algorithm. *Journal of Internet Services and Applications*, 15(1):495–507. DOI: 10.5753/jisa.2024.4044.

Cao, X., Zhang, Z., Jia, J., and Gong, N. Z. (2022). Flcert: Provably secure federated learning against poisoning attacks. *IEEE Transactions on Information Forensics and Security*, 17:3691–3705. DOI: 10.1109/tifs.2022.3212174.

de Souza, A. M., Maciel, F., da Costa, J. B., Bittencourt, L. F., Cerqueira, E., Loureiro, A. A., and Villas, L. A. (2024). Adaptive client selection with personalization for communication efficient federated learning. *Ad Hoc Networks*, 157:103462. DOI: 10.1016/j.adhoc.2024.103462.

Ghodsi, Z., Javaheripi, M., Sheybani, N., Zhang, X., Huang, K., and Koushanfar, F. (2023). zprobe: Zero peek robustness checks for federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4860–4870. DOI: 10.48550/arXiv.2206.12100.

Gu, C., Cui, X., Zhu, X., and Hu, D. (2023). Fl2dp: Privacy-preserving federated learning via differential privacy for artificial iot. *IEEE Transactions on Industrial Informatics*, 20(4):5100–5111. DOI: 10.1109/tii.2023.3331726.

Guo, Y. (2023). A review of machine learning-based zero-day attack detection: Challenges and future directions. *Computer communications*, 198:175–185. DOI: 10.1016/j.comcom.2022.11.001.

Hasan, N., Alam, M. G. R., Ripon, S. H., Pham, P. H., and Hassan, M. M. (2025). An autoencoder-based confederated clustering leveraging a robust model fusion strategy for federated unsupervised learning. *Information Fusion*, 115:102751. DOI: 10.1016/j.inffus.2024.102751.

Ma, X., Zhu, J., Lin, Z., Chen, S., and Qin, Y. (2022a). A state-of-the-art survey on solving non-iid data in federated learning. *Future Generation Computer Systems*, 135:244–258. DOI: 10.1016/j.future.2022.05.003.

Ma, Z., Ma, J., Miao, Y., Li, Y., and Deng, R. H. (2022b). Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning. *IEEE Transactions on Information Forensics and Security*, 17:1639–1654. DOI: 10.1109/tifs.2022.3169918.

Rezaei, H., Taheri, R., and Shojafar, M. (2025). Fedllmguard: A federated large language model for anomaly detection in 5g networks. *Computer Networks*, page 111473. DOI: 10.1016/j.comnet.2025.111473.

Smestad, C. and Li, J. (2023). A systematic literature review on client selection in federated learning. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, EASE '23, page 2–11, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3593434.3593438.

Song, R., Zhou, L., Lakshminarasimhan, V., Festag, A., and Knoll, A. (2022). Federated Learning Framework Coping with Hierarchical Heterogeneity in Cooperative ITS. In *IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. DOI: 10.1109/ITSC55140.2022.9922064.

Sun, P., Liu, X., Wang, Z., and Liu, B. (2024). Byzantine-robust decentralized federated learning via dual-domain clustering and trust bootstrapping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24756–24765. DOI: 10.1109/cvpr52733.2024.02338.

Veiga, R., Morais, R., Bastos, L., Rosário, D., Loureiro, A., and Cerqueira, E. (2025). A resilient and lightweight layer client selection in federated learning. In *2025 21st International Conference on Distributed Computing in Smart Systems and the Internet of Things*

(DCOSS-IoT), pages 609–616. IEEE. DOI: 10.1109/dcoss-iot65416.2025.00097.

Xia, G., Chen, J., Yu, C., and Ma, J. (2023). Poisoning attacks in federated learning: A survey. *Ieee Access*, 11:10708–10722. DOI: 10.1109/access.2023.3238823.

Yan, G., Wang, H., Yuan, X., and Li, J. (2023). Defl: defending against model poisoning attacks in federated learning via critical learning periods awareness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10711–10719. DOI: 10.1609/aaai.v37i9.26271.

Yazdinejad, A., Dehghantanha, A., Karimipour, H., Srivastava, G., and Parizi, R. M. (2024). A robust privacy-preserving federated learning model against model poisoning attacks. *IEEE Transactions on Information Forensics and Security*, 19:6693–6708. DOI: 10.1109/tifs.2024.3420126.

Ye, M., Fang, X., Du, B., Yuen, P. C., and Tao, D. (2023). Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys*, 56(3):1–44. DOI: 10.1145/3625558.

Zang, L. and Li, Y. (2024). Detection and mitigation of label-flipping attacks in fl systems with kl divergence. *IEEE Internet of Things Journal*, 11(19):32221–32233. DOI: 10.1109/jiot.2024.3424188.

Zhang, J., Hua, Y., Wang, H., Song, T., Xue, Z., Ma, R., and Guan, H. (2023). Fedala: Adaptive local aggregation for personalized federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11237–11244. DOI: 10.1609/aaai.v37i9.26330.

Zhao, K., Wang, L., Yu, F., Zeng, B., and Pang, Z. (2025). Fedmp: A multi-pronged defense algorithm against byzantine poisoning attacks in federated learning. *Computer Networks*, 257:110990. DOI: 10.1016/j.comnet.2024.110990.