Universidade Estadual de Londrina

BLENDA OLIVEIRA MAZETTO

A TRIAD OF DEFENSES TO MITIGATE POISONING ATTACKS IN FEDERATED LEARNING

LONDRINA
2025

BLENDA OLIVEIRA MAZETTO

A TRIAD OF DEFENSES TO MITIGATE POISONING ATTACKS IN FEDERATED LEARNING

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Bruno Bogaz Zarpelão

LONDRINA
2025

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Mazetto, Blenda Oliveira.
A Triad of Defenses to Mitigate Poisoning Attacks in Federated Learning / Blenda Oliveira Mazetto. - Londrina, 2025.
70 f. : il.

Orientador: Bruno Bogaz Zarpelão.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2025.
Inclui bibliografia.

1. Aprendizado Federado - Tese. 2. Ataques de Envenenamento - Tese. 3. Aprendizado de Máquina - Tese. I. Zarpelão, Bruno Bogaz. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU 519

BLENDA OLIVEIRA MAZETTO

A TRIAD OF DEFENSES TO MITIGATE POISONING ATTACKS IN FEDERATED LEARNING

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Bruno Bogaz Zarpelão
Universidade Estadual de Londrina

Prof. Dr. Sylvio Barbon Junior
Universidade degli Studi di Trieste – UNITS

Prof. Dr. Daniel Fernando Pigatto
Universidade Tecnológica Federal do Paraná – UTFPR

Londrina, 01 de Julho de 2025.

Este trabalho é dedicado às crianças adultas
que, quando pequenas, sonharam em se
tornar cientistas.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão ao meu orientador, Prof. Dr. Bruno Bogaz Zarpelão, pelo apoio contínuo ao longo dos últimos quatro anos, desde a iniciação científica até a conclusão deste mestrado. Sua orientação e incentivo foram fundamentais em cada etapa dessa jornada acadêmica.

Agradeço ao meu colega de laboratório, Filipe Viotto, pela contribuição durante a implementação de um dos ataques utilizados neste trabalho.

Agradeço também às minhas amigas Laura, Heloisa, Debora e Izabella, que estiveram sempre ao meu lado. Obrigada por me animarem nos momentos difíceis e por celebrarem comigo cada conquista. A amizade e o apoio de vocês foram essenciais para que eu chegasse até aqui.

À minha família, em especial à minha mãe, pelo apoio incondicional em todas as etapas desta jornada. Sua presença e incentivo foram fundamentais para que eu chegasse até aqui.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

"Não vos amoldeis às estruturas deste mundo, mas transformai-vos pela renovação da mente, a fim de distinguir qual é a vontade de Deus: o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2))

MAZETTO, B. O.. **Uma Tríade de Defesas para Mitigar Ataques de Envenenamento no Aprendizado Federado**. 2025. 70f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2025.

RESUMO

O Aprendizado Federado (AF) permite treinar modelos de Aprendizado de Máquina em dados descentralizados, promovendo a privacidade ao evitar o compartilhamento direto de dados entre clientes e servidor. No entanto, sua arquitetura distribuída o torna vulnerável a diversos tipos de ataques, com destaque para os ataques de envenenamento, devido à sua simplicidade e alto impacto no desempenho do modelo. Neste trabalho, propomos um método para mitigar esses ataques por meio de uma tríade de estratégias de defesa: organização dos clientes em grupos, verificação do desempenho local dos modelos globais durante o treinamento e uso de um esquema de votação na fase de inferência. Inicialmente, os clientes são organizados em grupos amostrados aleatoriamente. Cada cliente realiza o treinamento local de um modelo, e os modelos dentro de um mesmo grupo são combinados por meio de um processo de agregação, originando um modelo global distinto por grupo. Em seguida, cada cliente recebe e avalia todos os modelos globais, selecionando aquele com o melhor desempenho preditivo como o seu novo modelo local para continuar o treinamento. Por fim, durante a inferência, cada cliente realiza previsões sobre suas entradas com base em um esquema de votação entre os modelos globais. Os experimentos realizados com os conjuntos de dados MNIST, HAR e Air Quality demonstram que o método proposto é eficaz na mitigação de ataques de envenenamento, como Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling e Little Is Enough (LIE), preservando a integridade e o desempenho do modelo global.

Palavras-chave: Aprendizado Federado. Ataques de Envenenamento. Aprendizado de Máquina.

MAZETTO, B. O.. **A Triad of Defenses to Mitigate Poisoning Attacks in Federated Learning**. 2025. 70p. Master's Thesis (Master in Science in Computer Science) – State University of Londrina, Londrina, 2025.

ABSTRACT

Federated Learning (FL) allows training Machine Learning models on decentralized data, promoting privacy by avoiding the direct sharing of data between clients and the server. However, the distributed architecture of FL is vulnerable to various types of attacks, with poisoning attacks standing out due to their simplicity and high impact on model performance. In this work, we propose a method capable of mitigating such attacks through a triad of defense strategies: organizing clients into groups, evaluating the local performance of global models during training, and using a voting scheme in the inference phase. Initially, clients are organized into randomly sampled groups. Each client performs local training of a model, and the models within the same group are combined through an aggregation process, resulting in a distinct global model for each group. Then, each client receives and evaluates all global models, selecting the one with the best predictive performance as its new local model to continue training. Finally, during inference, each client makes predictions on its inputs according to a voting scheme among the global models. Experiments conducted with the MNIST, HAR, and Air Quality datasets demonstrate that our method is effective in mitigating poisoning attacks such as Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling, and Little is Enough (LIE), maintaining the integrity of the global model's results.

Keywords: Federated Learning. Poisoning Attacks. Machine Learning.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Traditional Machine Learning approaches require centralizing data on a single machine or datacenter. This data relating to users and organizations may contain private information that should not be shared, raising privacy concerns [1]. Federated Learning (FL) allows participants to collaboratively train a shared model while ensuring that all data remains stored locally on their devices, decoupling the ability to do Machine Learning from the need to store all data in a centralized server [2, 3]. Compared to centralized learning, FL significantly reduces server computation costs by outsourcing and parallelizing the training process. FL also is a promising paradigm to empower on-device intelligence and mitigate the privacy and scalability issues in IoT systems [4].

In the initial iteration of a Federated Learning training scheme, the central server generates a global model with randomly initialized weights and distributes it to all participating clients. Each client then trains the received model locally using its private dataset, resulting in a local model. These local models are then sent back to the server. Upon receiving all local models, the server applies a process known as aggregation to combine them into a new global model. This process typically consists of combining the parameters of the local models into a single model by computing a weighted average, reflecting each client's contribution. The process is repeated iteratively over multiple rounds until the training converges. Upon completion, the final global model is used by the clients to make predictions on new data during the inference phase [5].

While offering advantages in privacy and data locality, FL is exposed to a wide range of security threats. These include inference attacks, such as membership inference and model inversion, which aim to extract sensitive information from clients' data by analyzing the global model [6, 7]. Evasion attacks represent another threat, where adversaries craft specific inputs at inference time to induce misclassification [8, 9]. In addition, backdoor attacks can embed hidden behaviors into the global model that activate only under specific input patterns [10]. Other notable threats include free-rider attacks, in which some clients benefit from the training without contributing meaningful updates [11], and Sybil attacks, where a single adversary emulates multiple clients to increase its influence in the aggregation process [12, 13].

Among these threats, Byzantine attacks stand out for their potential to compromise the core training process. These encompass a broad class of adversarial behaviors where clients may act arbitrarily or deceitfully, aiming to disrupt model convergence or degrade performance [14, 15]. In the FL setting, this includes sending incorrect, inconsistent, or deliberately crafted model updates to the central server, thereby corrupting the global model.

Within the category of Byzantine attacks, poisoning attacks are particularly impactful and extensively studied. These attacks can be divided into two main types: in data poisoning, malicious clients inject harmful samples into their local datasets, whereas in model poisoning, they manipulate the gradients or model parameters before sending them to the server [16, 17]. Such actions can significantly degrade the accuracy and reliability of the resulting global model on unseen data.

In this work, we focus on poisoning attacks due to their simplicity, ease of implementation, and high potential for damaging the model's performance, even when only a small fraction of clients are compromised. Furthermore, their stealthy nature makes them especially challenging to detect and mitigate in practical FL scenarios.

Robustness against Byzantine attacks and the preservation of security and privacy in FL have been central research topics. Exploring the field of Byzantine robust aggregation, Xu et al. [2022] and Li et al. [2023] propose aggregation techniques to identify suspicious local models and enhance robustness. Another widely used method against poisoning attacks is model analysis. Che et al. [2022] include a scoring system to differentiate clients, an election strategy to select representatives, and a selection strategy for committee formation, fostering a collaborative and secure training environment. Also using a model analysis method, Jebreel et al. [2024] propose a fragmentation technique and, in addition, global and local reputation vectors to select trustworthy clients. Zhang et al. [2023], Cao et al. [2021], and Cao et al. [2022] organize clients into subgroups to ensure a robust scenario against the influence of malicious clients. Ultimately, Andreina et al. [2020] use a method based on performance evaluation as a defense strategy, exploring a unique characteristic of FL, the multiple private datasets.

Our proposed approach combines three techniques to mitigate poisoning attacks in FL: dividing clients into groups, checking global model performance on local datasets, and making inferences based on a voting scheme. Initially, the central server randomly divides the clients into groups. After the clients complete local training, they send their local models to the central server, which generates a global model for each group. The global models are subsequently distributed to all clients, ensuring that every participant in the Federated Learning process receives all the global models generated by the groups. Once the clients receive the global models, they evaluate them using their own data and select the model with the best predictive performance. This selected global model becomes the client's new local model. These steps are repeated until the training is completed. After the training phase, the inference phase relies on a voting method. Given an input, a client uses the global models to make predictions and combines them according to a consensus strategy appropriate for the task, for example by selecting the most frequent label in classification or computing a weighted average in regression. Our approach addresses both classification and regression tasks, and its effectiveness was validated through

experiments on three datasets: MNIST, HAR and Air Quality, under five types of poisoning attacks: Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling and Little is Enough (LIE).

The combination of these three techniques leverages their strengths to address issues that arise when they are applied individually. While each technique alone can reduce the influence of corrupted local models to some extent, their effectiveness decreases quickly as the number of malicious clients increases. By combining these three approaches, we develop a method that is more resilient to an increasing number of malicious clients while ensuring that both training and test data remain exclusively on the clients' devices throughout the process.

The remainder of this work is organized as follows. Chapter 2, presents the theoretical foundation needed for the comprehension of the work. In Chapter 3, we present our proposed approach to mitigate attacks on FL. We report on our experimental evaluation and results in Chapter 4. Finally, we conclude this work in Chapter 5.

2 FUNDAMENTAL BACKGROUND

2.1 Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that focuses on developing systems that can learn and improve from experience without being explicitly programmed. In essence, Machine Learning enables computers to analyze data, identify patterns, and make decisions or predictions based on that data [26]. Over the past decades, the field of ML has led to significant progress in advanced learning algorithms and efficient data preprocessing methods. These systems' ability to solve complex problems relies on analytical models that produce predictions, rules, answers, recommendations, or similar results [27].

Using statistical methods, algorithms are trained to make classifications or predictions. The learning process begins with data collection and preparation, which serves as input for the learning model. The model is then trained to recognize patterns in the data and extract relevant information for the task at hand. During training, the model adjusts its parameters to minimize the prediction error between the input data and the desired outputs. Once trained, the model can be used to make predictions or classifications with new input data. Learning methods can be categorized into supervised, unsupervised, and reinforcement learning [28, 29].

- **Supervised learning**: Requires a training dataset with examples of both the input data and labeled outputs or target values. These input-output pairs in the training set are used during the training process, where the algorithm leverages these pairs (x,y) to minimize the model's error by adjusting its parameters. After the model is trained, it can predict the target variable y for new, unseen data points based on input features x. Supervised learning can be further divided into regression problems, where a numeric value is predicted, and classification problems, where the outcome is a categorical class [27].

- **Unsupervised learning**: Occurs when the system is tasked with identifying patterns in data without any predefined labels or target outputs. In this case, the input data consists only of variables x, and the goal is to uncover inherent structures, such as grouping similar instances (clustering) or projecting high-dimensional data into a lower-dimensional space (dimensionality reduction) [30].

- **Reinforcement learning**: Instead of using input-output pairs, involves defining the system's current state, setting a goal, providing a list of possible actions and

their constraints, and letting the ML model learn to achieve the goal through trial and error to maximize a reward [31].

Depending on the task, Machine Learning algorithms can be grouped into different categories, such as Regression Models, Instance-based Algorithms, Decision Trees, Bayesian Methods, and Neural Networks. It is important to note that these categories are not mutually exclusive; for example, Decision Trees can be used for both regression and classification tasks. Among them, Neural Networks are particularly notable for their flexible structure, which allows them to be adapted to a wide range of problems [32].

2.2 Neural Networks

Neural Networks (NNs) are a type of Machine Learning algorithm inspired by the structure and functioning of the human brain. Unlike traditional computational methods, which rely on sequential steps to execute tasks, Neural Networks consist of interconnected nodes, or "neurons", capable of processing and analyzing vast volumes of data in parallel [33]. This architecture enables them to excel at tasks involving pattern recognition and complex decision-making.

The history of Neural Networks dates back to the 1940s and 1950s, when researchers began experimenting with artificial Neural Networks as a way to model and simulate the human brain [34]. In 1975, Kunihiko Fukushima introduced the concept of the multilayered Neural Network [35]. However, it was not until the 1980s and 1990s that Neural Networks gained significant traction.

The structure of a NN typically consists of three types of layers. The first is the input layer, which receives raw data from the dataset. For example, in an image recognition task, each neuron in the input layer might represent a pixel value from the image. Next are the hidden layers, where the data undergoes transformation through weighted connections and activation functions. These layers are responsible for capturing complex patterns in the data, such as edges in images or relationships between words in text. Finally, the output layer produces the network's final output, which could be a classification label (e.g., cat vs. dog) or a numerical prediction [36].

The training process of a Neural Network involves learning the optimal weights and biases to minimize prediction errors. Despite differences in internal computations across architectures, the overall supervised learning procedure generally follows these steps:

1. **Initialization of weights and biases**: Weights and biases are initialized with small random values, to ensure stable training.

2. **Forward propagation**: The Neural Network processes a set of input data through its layers. Each neuron computes the weighted sum of its inputs, adds a bias, and passes the result through an activation function to introduce non-linearity and capture complex patterns.

3. **Error calculation**: The error (or loss) is calculated by comparing the predicted output with the desired output using a predefined loss function.

4. **Backpropagation**: The error is propagated backward through the network using the chain rule to calculate gradients of the loss with respect to each weight and bias.

5. **Adjustment of weights and biases**: Using the gradients computed during backpropagation, the weights and biases are updated via an optimization algorithm. The learning rate determines the size of the updates.

6. **Repetition of the process**: Steps 2 to 5 are repeated for each batch of training data over n epochs.

Hyperparameters such as learning rate, batch size, and number of epochs play a crucial role in determining the speed and accuracy of the training process [37].

Over recent decades, advances in computational resources and the availability of large datasets have propelled Neural Networks to new heights. Researchers have developed increasingly accurate and sophisticated models that can be applied to a wide range of applications, including image recognition, speech recognition, and natural language processing.

2.2.1 Multi-Layer Perceptron Networks

Multi-Layer Perceptrons (MLPs) are a class of feedforward Artificial Neural Networks that consist of multiple layers of interconnected neurons. Each neuron performs a weighted sum of its inputs followed by a non-linear activation function. MLPs represent one of the earliest and most fundamental architectures in Machine Learning and have been widely applied to problems in classification, regression, and function approximation [38, 39].

Unlike simple linear models, MLPs can approximate complex non-linear functions, thanks to the use of activation functions such as the sigmoid, hyperbolic tangent, and Rectified Linear Unit (ReLU). The universal approximation theorem states that a feedforward network with a single hidden layer containing a finite number of neurons can approximate any continuous function on compact subsets of $\mathbb{R}^n$, under mild assumptions on the activation function [39].

Each layer in an MLP performs the following transformation:

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$
$$a^{(l)} = \phi(z^{(l)})$$

In this formulation, $W^{(l)}$ and $b^{(l)}$ represent the weight matrix and bias vector of the l-th layer, respectively. The term $a^{(l-1)}$ denotes the activation from the previous layer, $z^{(l)}$ is the result of the linear transformation, and $\phi(\cdot)$ is a non-linear activation function applied element-wise to the transformed input.

Despite the emergence of more specialized neural architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), MLPs remain a valuable baseline and are widely used in situations where data lacks spatial or temporal structure. Their versatility and simplicity have enabled their application across a wide range of domains, including speech recognition [40], financial forecasting [41], medical diagnosis [42], and image classification [43]. Moreover, MLPs form the foundation for more recent architectures, such as Transformer-based models, where feedforward components often resemble MLP layers [44], further demonstrating their continued relevance in modern Machine Learning systems.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a specialized class of Neural Networks designed to process data with a grid-like topology. Examples include time-series data, which can be viewed as a 1-D grid sampled over time, and image data, which forms a 2-D grid of pixels. Inspired by the visual cortex of animals, CNNs were first introduced by LeCun et al. in the 1980s and gained prominence with the success of the LeNet-5 architecture for handwritten digit recognition [45]. CNNs are particularly effective at spatial feature extraction, leveraging local connectivity and shared weights to efficiently learn hierarchical patterns from raw input data.

Unlike traditional fully connected networks, CNNs consist of convolutional layers that apply learnable filters to local regions of the input. Each filter, or kernel, computes a weighted sum over a local patch, producing a feature map that highlights specific patterns such as edges or textures. This local connectivity allows the network to exploit spatial structure and build hierarchical representations, where deeper layers capture increasingly abstract features [46].

After the convolution operation, a non-linear activation function, is applied to introduce non-linearity [47]. Pooling layers, are then used to reduce the spatial dimensions of feature maps and enhance translational invariance [48]. As the network deepens, the extracted features are flattened and passed to fully connected layers, which perform the final prediction.

The overall architecture of a CNN typically includes several stacked convolutional and pooling layers, followed by one or more fully connected layers for classification or regression. Regularization techniques such as dropout [49], batch normalization [50], and data augmentation are commonly used to improve generalization and mitigate overfitting.

CNNs have revolutionized computer vision, achieving state-of-the-art results in image classification [51], object detection [52], and image generation [53]. Despite the emergence of alternative architectures like Vision Transformers (ViTs) [54], CNNs remain a foundational component in Machine Learning for structured spatial data, offering efficient and interpretable mechanisms for feature extraction.

2.2.3 Long Short-Term Memory Networks

Long Short-Term Memory Networks (LSTMs) are a special type of Recurrent Neural Network (RNN) designed to handle sequential data and overcome the limitations of traditional RNNs. Introduced by Hochreiter and Schmidhuber in 1997 [55], LSTMs address a critical issue in RNNs known as the vanishing gradient problem, which hinders the network's ability to learn long-term dependencies.

Traditional RNNs are capable of learning from sequences by maintaining a hidden state that gets updated at each time step. However, as the length of the sequence increases, these networks struggle to retain information from earlier time steps. Gradients used for learning either vanish or explode during backpropagation through time, leading to poor performance when modeling long-range dependencies.

LSTM contains special units called memory blocks in the recurrent hidden layer. The memory blocks contain memory cells with self-connections storing the temporal state of the network in addition to special multiplicative units called gates to control the flow of information [56]. Each memory block in the original architecture contains an input gate and an output gate. The input gate controls the flow of input activations into the memory cell [57]. The output gate controls the output flow of cell activations into the rest of the network. Later, the forget gate was added to the memory block, proposed by Gers et al. to allow the network to reset its state [58]. By combining these three components, LSTMs can maintain and update their memory across long sequences in a controlled and effective manner. In addition, the modern LSTM architecture contains peephole connections from its internal cells to the gates in the same cell to learn precise timing of the outputs [59].

Mathematically, at each time step t, the LSTM processes the input vector x_t and the hidden state from the previous time step h_{t-1}. The following computations are performed:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && \text{(Forget gate)} \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && \text{(Input gate)} \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) && \text{(Candidate cell state)} \\
C_t &= f_t \times C_{t-1} + i_t \times \tilde{C}_t && \text{(Updated cell state)} \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) && \text{(Output gate)} \\
h_t &= o_t \times \tanh(C_t) && \text{(Hidden state output)}
\end{aligned}
$$

Here, σ represents the sigmoid activation function, which compresses the gate outputs to the range $[0, 1]$, and tanh is the hyperbolic tangent function, which helps in rescaling and controlling information flow. The cell state C_t acts like a conveyor belt, carrying relevant information across many time steps with minimal modification, thus making it easier for the network to retain long-term information [60].

One of the key strengths of LSTMs is their ability to learn context over varying time spans, making them suitable for a broad range of sequence modeling tasks. These include natural language processing, speech recognition, handwriting generation, and financial time-series forecasting [61].

Despite their complexity, LSTMs have proven to be robust and effective in practice. Over the years, they have been used to set performance benchmarks in various domains and continue to serve as a foundation for more advanced architectures, such as the Gated Recurrent Unit (GRU) and attention-based models like the Transformer.

2.3 Federated Learning

In Machine Learning, the training process often requires large amounts of data. This data, related to users and organizations, may contain private information that should not be shared without proper protection, raising privacy concerns. Federated Learning enables collaborative Machine Learning without the need to share data, only the learning model. Thus, all training data remains on local devices, and no individual updates are necessarily stored on a central or remote server [62].

Figure 1 presents a Federated Learning scheme, which typically involves clients (also referred to as workers) and at least one central server. In this setup, the clients are the owners of their local data, while the central server is responsible for generating a global learning model that can be shared and used by all participating clients.

The process begins with the central server distributing an initial global model to all participating clients. Each client trains this model locally using its own data, producing an updated version tailored to its dataset. These locally trained models are then sent back to

the server, which aggregates them to generate a new, improved global model. This updated model is redistributed to the clients, and the cycle continues over multiple rounds. With each iteration, the global model becomes increasingly accurate as it integrates knowledge from all clients, all while ensuring that raw data remains private.

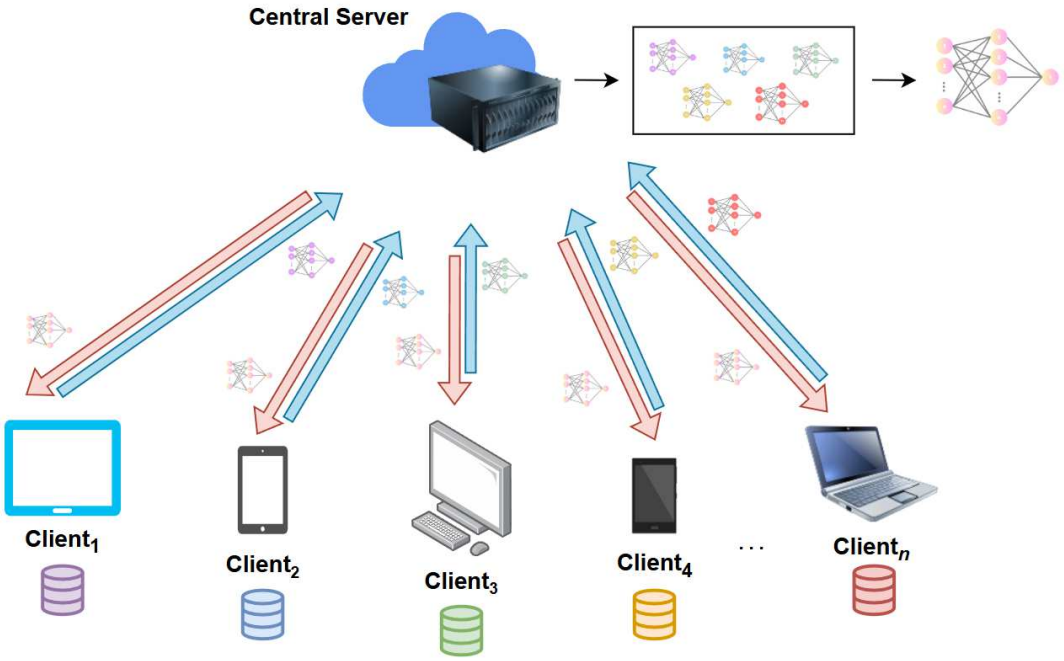

Figure 1 – Scheme illustrating the operation of a generic system based on Federated Learning.

The operation of Federated Learning can be described in the following steps, as illustrated in Figure 1:

1. Initially, the central server shares a global model with the clients, where the model weights are initialized with random values;
2. Each client trains the received model using its private dataset;
3. After training, all clients send their locally trained model to the central server;
4. The central server aggregates the received local models into a single global model;
5. The central server sends the aggregated model back to all clients for the next round of training.

Inference in Federated Learning follows a similar process to the training phase. Clients receive the global model and apply it to their local data to perform predictions. Depending on the task, this may involve classifying inputs or generating continuous outputs, allowing the model to support various applications across different domains.

A key challenge in Federated Learning is that client data is often non-IID, meaning it is not independently or identically distributed across participants. Unlike centralized settings, where data is usually IID, FL must deal with heterogeneous and personalized datasets. This can negatively affect the global model's performance, causing slower convergence, biased updates, and reduced generalization, while also making aggregation more complex and less robust.

The aggregation process plays a key role in Federated Learning for several reasons. It allows locally trained models from different clients to be combined into an updated global model. This is essential to enable local models to benefit from the collective knowledge. There are multiple forms of aggregation, next are some examples of different aggregation algorithms.

- FedAvg: FedAvg aggregation refers to the weighted averaging of local models in the context of FL, achieved by combining the Stochastic Gradient Descent (SGD) from each client and performing a weighted average of the models. The weighted average is calculated by taking into account the amount of data available on each device [63].

- FedProx: The goal of FedProx is to minimize the loss function (error function) of the Machine Learning model while, at the same time, limiting the impact of variation in training data across different clients. This is important because, in distributed systems, each client have a different set of training data, which can affect the quality of the final model. FedProx aims to balance the contribution of each client in updating the model parameters, thus avoiding bias towards a specific subset of data [64].

- Krum: Krum aggregation selects one of the m local models that closely resembles other models to serve as the global model. The idea is that, even if the chosen local model comes from a compromised client, its influence might be limited since it is close to other local models, likely from benign client devices. This method is specifically designed to mitigate poisoning attacks [65].

- Median: In the median aggregation method, for each jth parameter, the central server sorts the jth parameters of the m local models and selects the median as the jth parameter of the global model. If m is an even number, the median is the average of the middle two parameters. This method has been shown to provide robustness against poisoning attacks [65].

- Trimmed Mean: This aggregation method independently processes each model parameter. Specifically, for the jth parameter, the central server arranges the jth

parameters of the m local models (i.e., $w_{1j}, w_{2j}, ..., w_{mj}$, where w_{ij} is the jth parameter of the ith local model), removes the largest and smallest β parameters, and calculates the mean of the remaining $m - 2\beta$ parameters to determine the jth parameter of the global model. This technique is also designed to mitigate poisoning attacks [65].

Federated Learning is a general framework for collaborative model training across decentralized data sources. One of its key advantages is that it is not restricted to any specific type of algorithm. As long as a model supports parameter updates that can be aggregated, it can be adapted to the FL setting. In practice, FL has been successfully applied to a variety of Machine Learning algorithms beyond Neural Networks. For instance: Logistic Regression [66], Support Vector Machines (SVMs) [67], Decision Trees and Ensemble Methods [68], Naive Bayes [69], and Clustering algorithms such as k-means [70].

2.4 Vulnerabilities in Federated Fearning

Federated Learning introduces a novel paradigm for safeguarding user privacy while enabling large-scale Machine Learning tasks. However, the decentralized nature of FL also brings forth new vulnerabilities. Insider attacks are typically perpetrated by clients but can also originate from the server itself [17]. In contrast to traditional Machine Learning, attackers in FL systems can assume various forms, including:

- Clients: The server lacks control over participants' behaviors, allowing a malicious participant to deviate from the established training protocol and compromise the global model. When an adversary gains control over a client, they can execute actions such as monitoring the global model, corrupting or substituting local model updates (known as poisoning attacks), and manipulating the training process, including the optimization of the loss function and hyperparameters [71].
- Central Server: An adversary in control of the central server can directly inspect and alter the parameters of the global model, as well as scrutinize all local model updates from the clients. Additionally, honest-but-curious or semi-honest server adversaries may attempt to infer private information from the model updates received during the execution of the protocol [72].
- Outsiders or Eavesdroppers: As an external threat, an adversary can intercept communications between participants and carry out inference-time attacks [73].

Among the attacks outlined above, FL is particularly vulnerable to poisoning attacks, which are malicious manipulations of local data or model, due to its inherently

permissionless design, where any client can participate in the training process. However, the risk is not limited to open systems; even in controlled or authenticated environments, compromised clients may behave maliciously or unpredictably by submitting poisoned updates, thereby undermining the integrity of the global model.

2.4.1 Poisoning Attacks in Federated Learning

A poisoning attack in Federated Learning occurs when an attacker alters the model uploaded by a client to the central server during the aggregation phase, either directly or indirectly, resulting in an incorrect update to the global model. These attacks can be categorized based on the method the attacker employs to modify the local model parameters, leading to the generation of a poisoned model (model poisoning attacks and data poisoning attacks). Additionally, poisoning attacks can be classified according to the attacker's intent (targeted, semi-targeted, and untargeted poisoning attacks) [14].

The following types of attacks are classified according to the poisoning attack method, as illustrated in Figure 2:

- Data Poisoning Attack: They are primarily divided into two types: clean-label attacks [74] and dirty-label attacks [75]. Clean-label attacks involve modifying samples in the training set, such as by introducing noise into the training data. Furthermore, an attacker may alter the samples in a specific manner to implant a backdoor in the global model. Conversely, executing dirty-label poisoning requires the adversary to introduce multiple copies of data samples they wish to misclassify, assigned with a target label, into the training set. In such scenarios, there is no certification process to verify that a data sample belongs to the correct class. A common example of a dirty-label poisoning attack is label-flipping. Data poisoning attacks in FL mainly focus on dirty-label poisoning since FL operates under the premise that data is never shared, only learned models. Consequently, the adversary is not concerned with issues of imperceptibility for data certification.

- Model Poisoning Attack: These attacks target the local training process in Federated Learning by manipulating model updates, such as altering gradients directly. To evade detection, attackers exploit their control over the local training process, strategically adjusting their updates. They aim to optimize both the training loss and an adversarial objective, using parameter estimation to ensure their updates closely align with those of benign clients. This tactic helps avoid noticeable discrepancies in the global model, making the malicious behavior harder to detect. Generally speaking, model poisoning attacks are far more effective than data poisoning in FL contexts. A single, non-colluding malicious participant can lead the global model to misclassify a specific set of inputs with high confidence. This occurs

because the updates from the malicious client can be often amplified and tailored to inflict maximum damage on the performance of the global model [76].

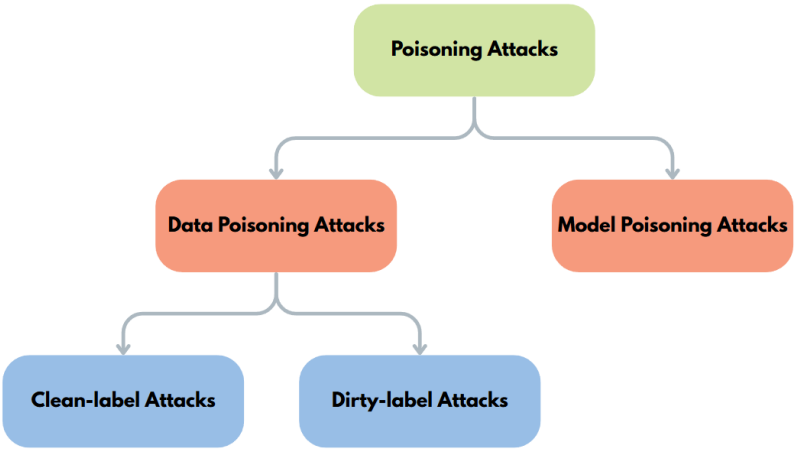

Figure 2 – Taxonomy of poisoning attacks in Federated Learning based on the attack method.

The types of attacks are now organized according to the attacker's intent, as illustrated in Figure 3:

- Targeted Poisoning Attack: In a targeted poisoning attack, the attacker aims to specifically degrade the performance of the global model on a designated task while leaving other tasks unaffected. The targeted test inputs may consist of specific test inputs, inputs with certain characteristics, or inputs embedded with a particular trigger. If the targeted test inputs are those that contain a trigger, the attack is referred to as a backdoor attack. The objective of a backdoor attack is to corrupt the global model so that it predicts the attacker-chosen target label for any test input that includes the predefined trigger [71].

- Semi-targeted Poisoning Attack: In a semi-targeted attack, the attacker picks a specific class (the source class) and aims to poison the global model such that samples from this source class are misidentified as a different class. Unlike in targeted attacks, the attacker has the flexibility to select the target class to optimize the effectiveness of the semi-targeted attack [14].

- Untargeted Poisoning Attack: The objective of untargeted poisoning attacks is to reduce the test accuracy of the learned global model. These attacks aim to maximize the indiscriminate error rate of the global model. The simplest approach to achieve this goal is by introducing random noise to the local model [77, 78].

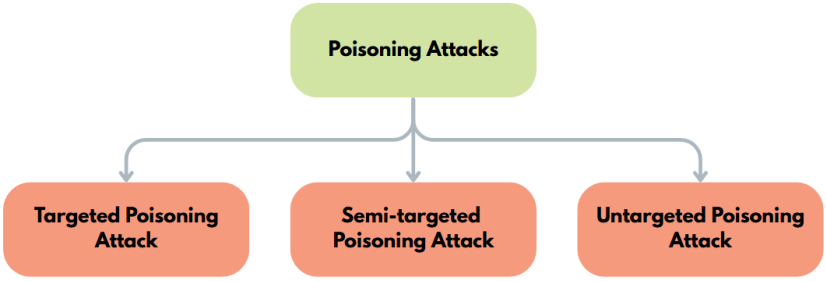

Figure 3 – Taxonomy of poisoning attacks in Federated Learning based on the attacker's intent.

2.5 Related Work

In the FL field, robustness against Byzantine attacks and preservation of security and privacy have been key focus areas. Various methods and frameworks have been proposed to mitigate these threats and ensure the integrity of the globally trained models. According to Xia et al. [2023], defense strategies against poisoning attacks can be divided into three categories: 1) Model analysis, 2) Byzantine robust aggregation, and 3) Verification-based methods. Model analysis methods operate under the assumption that significant differences exist between poisoned and benign models, and that these differences can be distinguished. In response, the Byzantine robust aggregation strategy serves as a passive defense mechanism, mitigating the impact of poisoning attacks by altering the global model's aggregation method. Complementing this, the Verification-based defense strategy further strengthens security by introducing a verification step, which prevents attackers from forging data or models and complicates the execution of attacks.

Within the category of Byzantine robust aggregation defense, Xu et al. [2022] propose a filtering strategy based on Truth Discovery aggregation, an unsupervised iterative technique that identifies and eliminates unreliable local updates. Their approach, TDFL, demonstrates strong robustness even in Byzantine-majority scenarios (>50% malicious clients), effectively mitigating attacks without relying on a validation set or reference model. However, the method does not address incentive mechanisms for honest participation, which is noted as a limitation to be explored in future work. Complementing this category, Li et al. [2023] propose AutoGM, a secure aggregation rule, variant of the Geometric Median that adaptively excludes outliers and reweights updates based on skewness thresholds. AutoGM can be applied in both traditional FL paradigms and Personalized FL paradigms, demonstrating strong robustness against model and data poisoning attacks. However, the approach depends on user-defined hyperparameters, which may require careful tuning to maintain performance across varying conditions.

Expanding on this category, foundational methods such as Krum, Median, and Trimmed Mean have played pivotal roles in countering Byzantine threats. Blanchard et

al. [2017] introduced Krum, an aggregation rule that selects a local model closest to the majority of other models, effectively filtering out adversarial outliers. Yin et al. [2018] proposed both Trimmed Mean and Median, each offering distinct strategies to enhance robustness. Trimmed Mean aggregates model parameters independently, discarding a fixed proportion of the highest and lowest values in each dimension, balancing computational efficiency and resilience to adversaries. Median computes the element-wise median of all updates, reducing the influence of extreme values and providing strong defense against malicious attacks. However, more recent studies have shown that tailored attacks can be designed to exploit the vulnerabilities of these aggregation rules, reducing their effectiveness in adversarial settings and highlighting the need for more adaptive or context-aware defenses.

Moving towards a decentralized approach, Che et al. [2022] explore the model analysis defense strategy, presenting CMFL, a serverless FL framework that employs a committee mechanism. In this framework, some clients are elected as committee members responsible for monitoring the training process and ensuring reliable aggregation of local gradients. CMFL introduces a scoring system to evaluate clients, an election strategy to select representatives, and multiple committee selection strategies tailored to different scenarios. The framework achieves faster convergence and improved model performance compared to traditional and Byzantine-tolerant FL models. However, its robustness primarily relies on detecting abnormal gradients, and it remains vulnerable to targeted attacks such as backdoor poisoning, which highlighs the need for more advanced election and selection mechanisms with formal guarantees under such conditions.

Continuing with the model analysis strategy, aimed at enhancing security and privacy, Jebreel et al. [2024] propose a novel lightweight protocol that enables participants to privately exchange and mix random fragments of their model updates before submission to the server. This design preserves the coordinate positions of parameters, allowing accurate aggregation while preventing the server from reconstructing original updates or linking them to specific users. The approach is reinforced by a reputation-based mechanism, where both global and local reputations guide participant selection and update weighting, improving resilience against adversarial behavior. However, the framework has not yet been evaluated under backdoor attacks or non-IID data scenarios, which can be critical challenges in practical Federated Learning deployments.

Another widely used defense strategy in recent years involves the possible groupings of clients. In this context, Zhang et al. [2023] organize clients into subgroups with a hierarchical k-ary tree structure, using random partitioning and partial parameter disclosure to limit attacker influence. They propose SAFELearning, a secure aggregation protocol that detects backdoor and model-poisoning attacks even over encrypted updates, while preserving model privacy. The protocol also improves scalability in computation

and communication.

Cao et al. [2021] propose an ensemble Federated Learning approach that uses majority voting among multiple global models trained on subsets of clients, also employing the defense strategy based on dividing clients into groups. This method ensures robustness when the majority of clients are honest, even in the presence of a limited number of malicious clients. Their approach achieves certified accuracy of 88% on MNIST when 20 out of 1,000 clients are malicious. Similarly, Cao et al. [2022] extend the method by grouping clients into probabilistic or deterministic subgroups, with each global model trained on a subgroup. Their final aggregation combines predictions from all models, enhancing robustness against malicious influence. The proposed FLCert framework provides provable security against poisoning attacks by using ensemble models from client groups, ensuring that the majority vote remains unaffected by malicious clients. However, a limitation of this work is that it does not incorporate prior knowledge about the learning task or the base FL algorithm when deriving certified security levels, which may affect its generalizability to diverse scenarios.

Finally, Andreina et al. [2020] propose a defense strategy that leverages a unique characteristic of FL (clients' access to private datasets) to detect backdoor attacks through performance evaluation.The authors propose BaFFLe, utilizing validation clients to detect if the global model update has been compromised by poisoning attacks, and discarding such updates when necessary. The results obtained from BaFFLe can achieve a detection accuracy of 100% with a false-positive rate below 5%, on both CIFAR-10 and FEMNIST datasets, even with small validation sets or when activated late in training. The defense remains compatible with secure aggregation protocols and requires minimal changes to existing FL setups. However, the method's effectiveness relies on the assumption that validation clients behave honestly and consistently hold representative data, which may not always apply in practice.

The approach proposed in this work combines three defense techniques against poisoning attacks. Similarly to what Cao et al. [2021] and Cao et al. [2022] propose, the first step of our approach is a probabilistic grouping division. The next technique we used for attack mitigation is model performance evaluation. Andreina et al. [2020] propose a strategy where clients' private datasets are used to verify if an attack has compromised the global model. In our approach, each client receives the global models and uses their private dataset to evaluate these models. After the evaluation, each client selects the global model with the best predictive performance to be their new local model. Finally, this work uses a voting strategy for inference, similar to the ensemble Federated Learning used by Cao et al. [2021] and Cao et al. [2022]. During the final step of our approach, each client receives the global models and determines the final output based on their combined responses, according to the nature of the task. The objective of combining these three strategies

is to enhance the model's resilience against an increasing number of malicious clients, addressing the challenge that other methods face when dealing with large proportions of compromised clients. Table 1 compares the reviewed studies and the proposed approach.

Table 1 – Comparison among the reviewed approaches based on their reliance on client grouping strategies, whether the central server allows different types of aggregation, and the use of client feedback for evaluating model performance. It also lists the types of attacks simulated in each study.

Related work	Uses grouping	Allows different aggregations	Uses client feedback	Simulated Attacks
[18]	×	×	×	Label-Flipping, Arbitrary Model, Krum, Trim and Backdoor
[19]	×	×	×	Label-Flipping and Gaussian
[20]	×	✓	✓	Malicious Gradients
[21]	✓	×	✓	Label-Flipping and Gaussian
[25]	×	×	✓	Label-Flipping
[22]	✓	×	×	Label-Flipping and Adaptive Semantic
[23]	✓	✓	×	Malicious Gradients
[24]	✓	✓	×	Label-Flipping, Same-Value, Krum and Trim
Our Approach	✓	✓	✓	Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling and LIE

3 PROPOSED APPROACH

Unlike traditional FL models, our proposed method begins with the random grouping of n clients into N groups of k clients each (these variables are commonly used in the FL literature). The purpose of sampling the clients into groups is that, as long as we do not have a vast majority of malicious clients, we still have a high chance of retaining uncompromised groups. When most clients are benign, the influence of malicious clients is reduced, as a malicious client can only affect the groups to which it belongs. It is also important to highlight that the random division of groups is done in a way that a client can belong to more than one group. Each client is assigned to each group independently with probability $p = \frac{k}{n}$, with the probability of a client belongs to more than one group given in Equation 3.1. Accordingly, when a client is assigned to multiple groups, its local model update is aggregated into the global model of each group to which it belongs. Figure 4 shows the division of 5 clients ($n = 5$) into 3 groups ($N = 3$), with each group containing 2 clients ($k = 2$). Randomly, clients 1 and 4 were assigned to group 1, clients 3 and 5 to group 2, and finally, clients 2 and 4 to group 3.

$$P(X \geq 2) = 1 - (1-p)^N - N\,p\,(1-p)^{N-1} \tag{3.1}$$

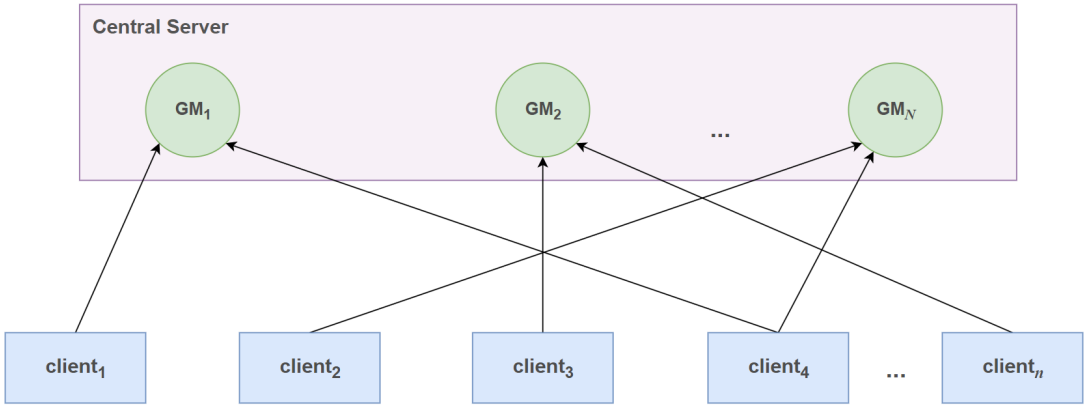

Figure 4 – Example of the group division process with $n = 5$, $N = 3$ and $k = 2$.

Once the groups are defined, the training is initiated. The central server sends a learning model to all clients, with this initial model having automatic weights that follow a uniform distribution, which will be updated over the training process. After receiving the model, the clients update it using their local data, thus generating n local models. Then, the clients send their local models to the central server.

The central server uses the groups defined earlier to aggregate the local models. Each one of the N groups generates a global model GM_i, which is the result from the

aggregation of the local models of the clients belonging to that group. Thus, we will have $GM_1, GM_2, GM_3, ..., GM_N$. This process can be observed in Figure 5 and detailed in Algorithm 1. Aggregation is an important step in FL systems. Our approach makes it possible to choose any aggregation method, as this does not affect the functioning of our method. After the global models are computed, the central server sends them to each client, a step that can be seen in Figure 6.

Algorithm 1 Group-based aggregation of local models by the central server

Input:
- *local_models*: List of models, where the i-th entry corresponds to client i
- *groups*: List of groups, each containing indices of participating clients

Output:
- *global_models*: List of global models, one per group

Server executes:

global_models ← [ ]
for each *group* in *groups* **do**
    group_models ← [ ]
    for each *client_index* in *group* **do**
        model ← *local_models[client_index]*
        append *model* to *group_models*
    end for
    Compute *aggregated_model* from *group_models* using chosen aggregation method (e.g., FedAvg)
    Append *aggregated_model* to *global_models*
end for
return *global_models*

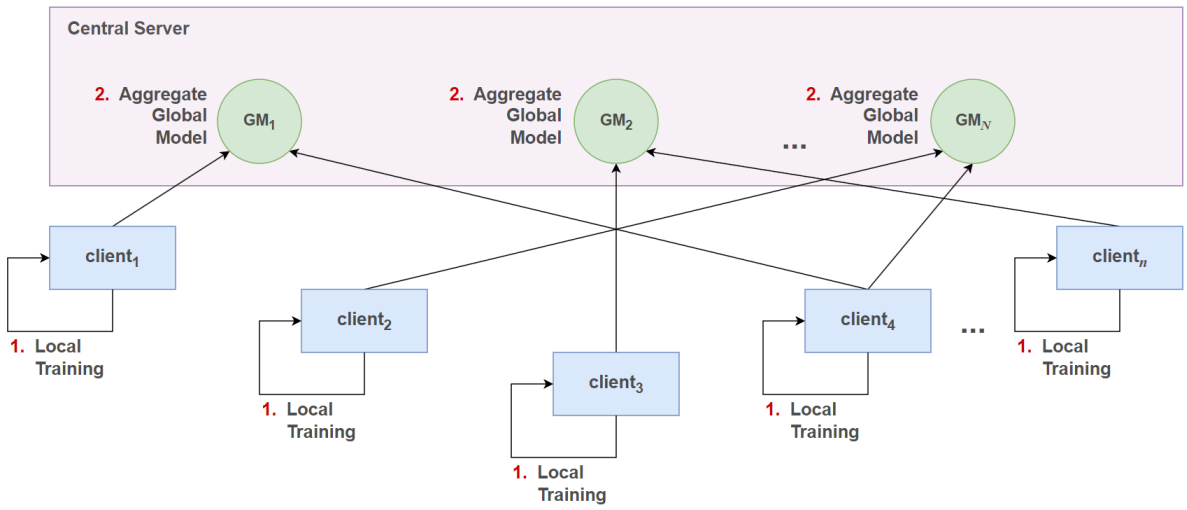

Figure 5 – First and second steps of our proposal, where we can visualize the clients training a local model with their private dataset and the central server aggregating the local models into the global models according to the group division.

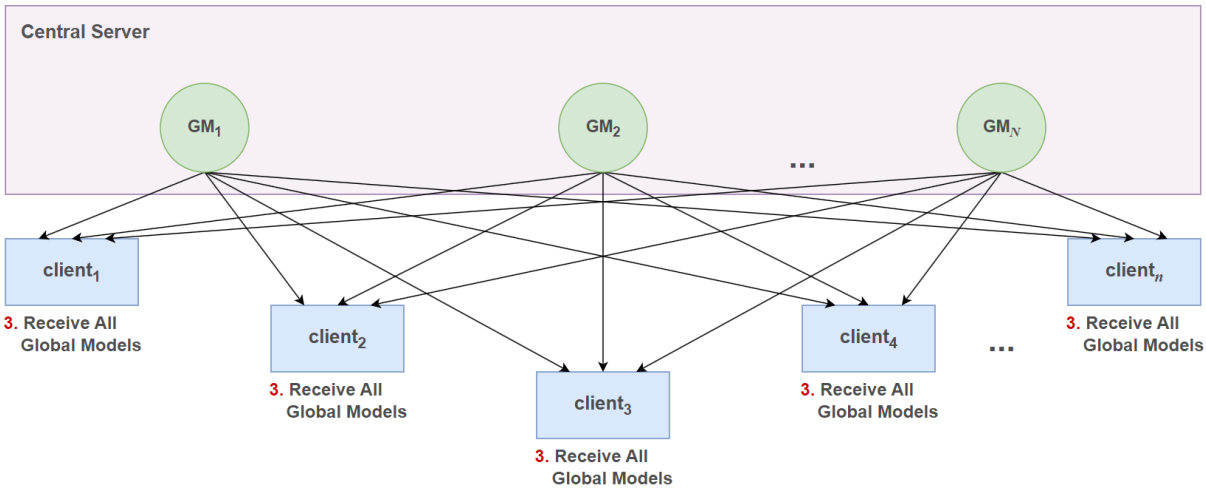

Figure 6 – Third step of our approach, where each client receives all previously calculated global models.

Sequentially, the proposed approach carries out a performance evaluation step, which aims to improve the whole system performance using clients' private datasets $D_1, D_2, ..., D_n$. Once the clients receive all the global models, they use their private validation datasets D_i to evaluate the global models $GM_1, GM_2, ..., GM_N$. Then, each client computes the evaluation metric for each global model based on their own data. In classification scenarios, the evaluation metric used is the F1-score, which is computed using the counts of true positives (TP), false positives (FP), and false negatives (FN) extracted from the confusion matrix, as defined in Equation 3.2. F1-score is a global metric for evaluating predictive performance, particularly useful in scenarios with class imbalance. In regression scenarios, the evaluation metric adopted is the Mean Absolute Error (MAE), which measures the average absolute difference between the predicted values $\hat{y}_i$ and the actual values y_i, as shown in Equation 3.3. MAE is a widely used metric for regression tasks, offering an intuitive measure of prediction accuracy by capturing the average magnitude of errors in a model's predictions, without considering their direction. Next, each client selects the global model that achieved the best score in the evaluation process and this global model becomes the new local model for that client, as shown in Figure 7 and outlined in Algorithm 2.

Global models produced by compromised groups are expected to perform worse in terms of evaluation metrics, since they were affected by poisoned models. This expectation is based on the nature of the poisoning attacks addressed by our approach, which are specifically designed to degrade model performance. As a result, clients tend to discard these compromised models during evaluation. By systematically avoiding underperforming global models during the performance evaluation process, the influence of poisoned models is gradually reduced over successive training rounds, as they are less likely to be chosen

and propagated. This dynamic acts as a filtering mechanism that helps suppress the long-term impact of adversarial behavior, contributing to the disappearance of poisoning effects on the global model. Furthermore, this solution allows us to use the clients' private datasets to guide model selection, without compromising data privacy.

$$F1 = \frac{2\,TP}{2\,TP + FP + FN} \tag{3.2}$$

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m} |\hat{y}_i - y_i| \tag{3.3}$$

Algorithm 2 Performance evaluation and selection of the best global model by the client

Input:
- *global_models*: List of global models, one per group
- *validation_data*: Client's private validation dataset
- *is_classification*: Boolean indicating the type of task

Output:
- *selected_model*: Global model with the best evaluation score

Client executes:

```
if is_classification then
    best_score ← −∞
else
    best_score ← ∞
end if
selected_model ← None
for each model in global_models do
    score ← EvaluateMetric(model, validation_data)
    if is_classification and score > best_score then
        best_score ← score
        selected_model ← model
    else if not is_classification and score < best_score then
        best_score ← score
        selected_model ← model
    end if
end for
return selected_model
```

The steps described so far are repeated until the end of the training. When the training is completed, we move to the inference phase, which relies on a voting method. Similarly to the previous steps, the clients' local models are aggregated according to the initially defined groups. Shortly after, the global models are sent to all clients. Once the clients have received all the global models, they start the voting step, where each client makes inferences with their own data. In the case of classification tasks, during the inference phase, the N global models are used to predict labels for inputs. Specifically,

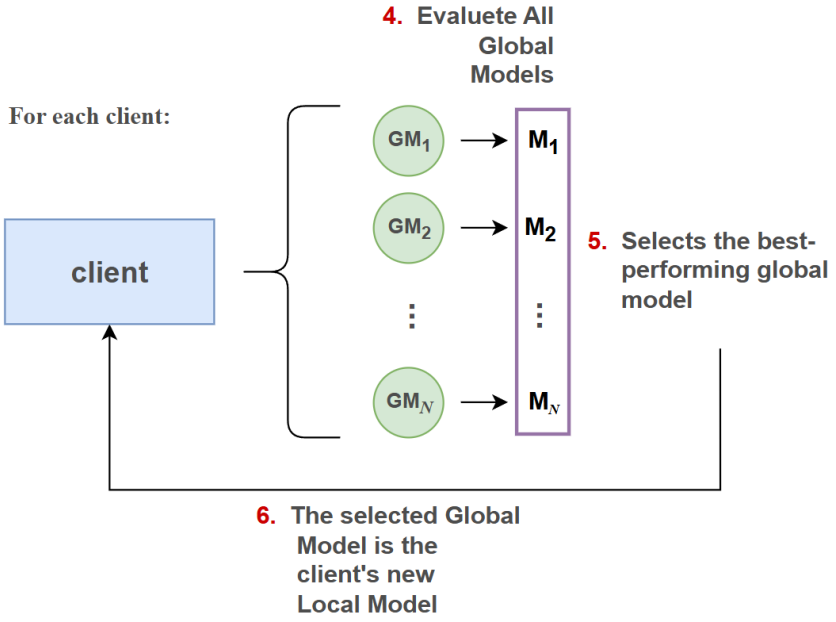

Figure 7 – Fourth, fifth and sixth steps of our approach, where each client evaluates the received global models and selects the best of them to become their new local model.

given a test input x, the client uses each global model to predict its label. After that, the client calculates the frequency of all predicted labels, which is the number of global models that predict a certain label for x. Thus, the client takes a majority vote among the N global models to predict the label for the input x. The label with the highest number of predictions is the resulting label. A detailed step-by-step description of this process is provided in Algorithm 3, complementing the illustration in Figure 8. The aim of this step is to ensure that the resulting label from the majority vote among the N global models remains unaffected by a limited number of malicious clients. When there are ties, i.e., multiple labels have the same highest frequency, the client randomly selects one of the tied labels.

In regression tasks, the inference procedure is adapted to reflect the continuous nature of the output. Instead of a voting mechanism based on majority vote, we apply a weighted average over the predictions of the global models. To determine the weights, each client uses the performance evaluation step described previously to score and rank the global models using its private validation dataset. These scores, based on MAE, are then transformed into weights that reflect the relative reliability of each model, and models with lower error receive higher weight. During inference, each global model receives the test input x and produces a prediction $\hat{y}_j$. The final output is computed as a weighted average of these predictions, where the weight of each prediction corresponds to the evaluation score of the respective global model. This approach ensures that better-performing models contribute more significantly to the final prediction, improving robustness and accuracy

in the presence of unreliable models. A detailed description of this process is provided in Algorithm 4 and illustrated in Figure 9.

Algorithm 3 Inference using majority voting over global models

Input:
- *global_models*: List of global models, one per group
- x: Test input

Output:
- *final_label*: Predicted label for x via majority voting

Client executes:

Initialize *label_votes* as an empty map (label → count)
for each *model* in *global_models* **do**
    label ← *model*.predict(x)
    Increment count for *label* in *label_votes*
end for
Identify label(s) with the maximum vote count
if there is a tie among top labels **then**
    final_label ← randomly select one of the tied labels
else
    final_label ← label with the highest vote
end if
return *final_label*

Algorithm 4 Inference using weighted average over global models

Input:
- *global_models*: List of global models, one per group
- x: Test input
- *model_scores*: Evaluation scores for each model (e.g., MAE)

Output:
- *final_prediction*: Predicted value for x via weighted average

Client executes:

Compute weights by inverting and normalizing *model_scores*
Initialize *weighted_sum* ← 0
Initialize *total_weight* ← 0
for each *model*, *weight* in *global_models*, *weights* **do**
    prediction ← *model*.predict(x)
    weighted_sum ← *weighted_sum* + *weight* × *prediction*
    total_weight ← *total_weight* + *weight*
end for
final_prediction ← *weighted_sum* ÷ *total_weight*
return *final_prediction*

3.1 Dealing with Malicious Selections

For classification tasks, during the performance evaluation phase of our method, each client assesses the global models $GM_1, GM_2, \ldots, GM_N$ using its private validation

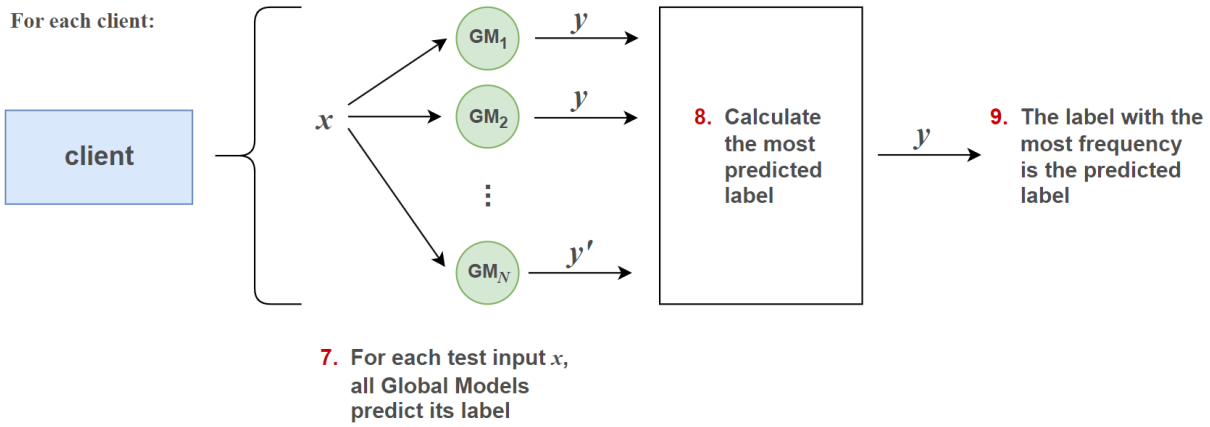

Figure 8 – Seventh, eighth and ninth steps of our approach, where each client makes inferences for their test data taking into account the majority vote among global models (for classification tasks).

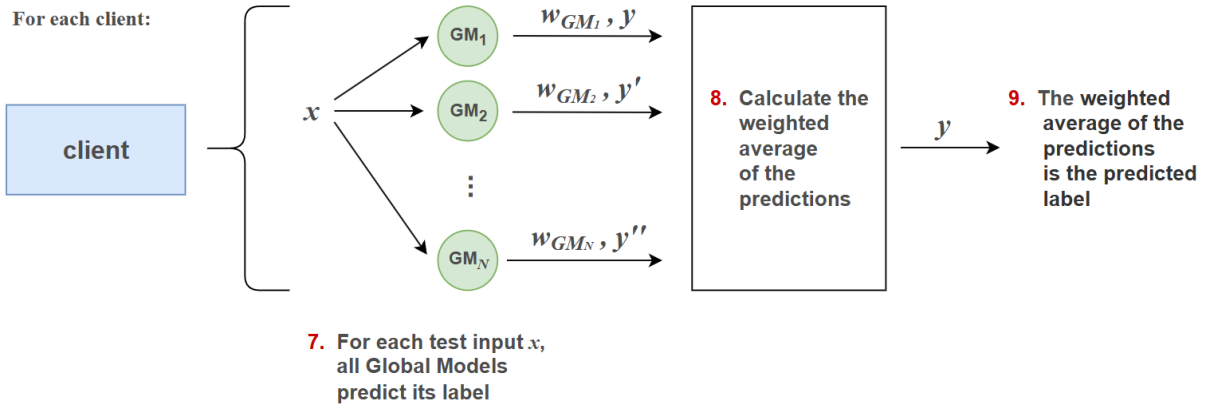

Figure 9 – Seventh, eighth and ninth steps of our approach, where each client makes inferences for their test data taking into account the weighted average among global models (for regression tasks).

dataset and selects the one with the highest F1-score as its new local model. A malicious client, however, may attempt to compromise the system by selecting the worst-performing global model, i.e., the one with the lowest F1-score, instead of the best, to degrade overall system performance.

Despite this threat, the system demonstrates resilience to such adversarial behavior as long as the number of malicious clients m remains less than half of the total number of clients n, i.e., $m < \frac{n}{2}$. This resilience stems from the majority voting mechanism used during inference. Each client uses all N global models to predict the label of a test input x, and the final output label is determined by majority vote over the predictions from these models. Even if some global models are influenced by malicious clients during training, they are likely to be outvoted if most global models were selected and trained by benign

clients who based their choices on accurate model evaluations.

This approach aligns with principles from Byzantine Fault Tolerance (BFT) theory, where systems that rely on majority consensus can tolerate up to $m < \frac{n}{2}$ malicious participants without compromising correctness [82]. While classical BFT requires $m < \frac{2}{3}$ for full consensus (as in blockchain or secure databases), systems based on simple majority voting can remain correct as long as fewer than 50% of participants are malicious. Empirical results supporting this claim are presented in Section 4.2.4. The experiments show that model performance remains stable until approximately 50% of the clients are malicious. When this threshold is exceeded, a significant drop in accuracy and F1-score is observed, confirming the expected breakdown point.

4 EXPERIMENTAL SETUP AND RESULTS

4.1 Experimental Setup

4.1.1 Datasets

We use the MNIST, HAR and Air Quality datasets for the experiments. Next, more details about them are presented:

- MNIST: The MNIST dataset [83] is a widely used dataset in Machine Learning, comprising 70,000 grayscale images of handwritten digits (0-9), each sized 28×28 pixels. Given its popularity for training Machine Learning models, we employed it to simulate FL scenarios. Our experiments were conducted with 30 and 50 clients. Initially, the dataset was split into training, validation, and test sets, with 50,000 samples for training, 10,000 for validation, and 10,000 for testing. In our federated environment, these subsets were evenly distributed among the clients, simulating each client having its own private dataset.

- HAR: The Human Activity Recognition Using Smartphones (HAR) dataset [84] was created from recordings of daily activities performed by individuals carrying a smartphone on their waist, which was equipped with inertial sensors. The experiments involved 30 volunteers aged 19 to 48, each performing six activities corresponding to the six labels in the dataset (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING). The dataset includes 561 features and 10,299 instances. The HAR dataset is naturally federated for 30 clients, since the data for each volunteer can be easily converted to the private dataset for a client. For this reason, HAR clients do not have the same number of samples (since the volunteers did not produce the same amount of samples). Therefore, the first step was to partition the dataset into private datasets for the clients, followed by splitting these private datasets into training, validation, and test sets based on percentages: 70% for training, 10% for validation, and 20% for testing.

- Air Quality: The Air Quality dataset [85] contains measurements of various air pollutants and meteorological variables collected from an urban monitoring station. It includes hourly averaged responses from gas sensors, along with corresponding temperature and humidity values, over several months. This dataset is well-suited for regression tasks due to its multivariate time-series nature. For our experiments, the dataset was divided into 10 sequential parts to preserve temporal order. Each part was assigned to a different client, resulting in 10 clients in total. The private

data for each client was then split into training, validation, and test sets, using 70%, 10%, and 20% of the data, respectively.

4.1.2 FL setup and Model Parameters

FL setup: For the MNIST dataset, experiments were conducted with two variations in the number of clients: $n = 30$ and $n = 50$. For the 30-client scenario, one variation of N were tested: $N = 15$. In the 50-client variation, three variations of N were also tested: $N = 15$, $N = 25$, and $N = 35$. For the HAR dataset, experiments were conducted with 30 clients, as the dataset is naturally federated for 30 clients. Three variations of N were tested: $N = 9$, $N = 15$, and $N = 21$. Regarding the number of clients per group, values closely aligned with those reported in the literature were selected. Thus, the scenarios mentioned above were tested with 3 and 5 clients per group ($k = 3$ and $k = 5$). For the Air Quality dataset, experiments were conducted with $n = 10$ clients. Three variations of N were tested: $N = 3$, $N = 5$, and $N = 8$. For each group, the number of clients per group was set to $k = 2$, $k = 3$, and $k = 4$. The aggregation method chosen was FedAvg, which calculates the average of local models. This method was selected for its performance, efficiency, and scalability potential. Additionally, compared to other aggregation methods like Krum, Trimmed Mean, and Median, FedAvg has reduced operational costs.

Model Architectures and Parameter Settings: For the MNIST dataset, we used a Convolutional Neural Network (CNN) architecture proposed by [23]. Key parameters included a batch size of 32, a learning rate of 0.001, and Stochastic Gradient Descent as the optimizer. The number of epochs was set to 100, with 10 global iterations. For the HAR dataset, we employed a Deep Neural Network (DNN) with two fully connected hidden layers, each containing 256 neurons and using ReLU activation functions, an architecture proposed by [23]. The parameters for this model included a batch size of 64, a learning rate of 0.001, and Stochastic Gradient Descent as the optimizer. The number of epochs and global iterations were 200 and 20, respectively. For the Air Quality dataset, we employed a Long Short-Term Memory (LSTM) network consisting of two layers with 64 hidden units each and a dropout rate of 0.2. The model output is produced through a fully connected layer. Training was performed with a batch size of 32, a learning rate of 0.001 using the Adam optimizer, for 100 epochs and 10 global iterations.

4.1.3 Performed Attacks

To evaluate the impact of poisoning attacks in a Federated Learning setting, we implemented a simulated FL environment composed of multiple clients and a central server, reflecting the typical architecture found in the literature. To simulate adversarial behavior, we incorporated poisoning attacks directly into the communication loop of the

system. Specifically, when a client was designated as malicious, its local behavior was altered programmatically: either by manipulating its training data, or by modifying the model updates before transmission. These manipulations were seamlessly integrated to preserve the protocol's flow, ensuring that the malicious clients remained indistinguishable from honest ones from the server's perspective.

We selected five poisoning attacks with distinct strategies: one targeting the data (Label-Flipping Attack), and four targeting the model (Same-Value Attack, Gaussian-Noise Attack, Gradient-Scaling Attack, and LIE Attack). While the Same-Value Attack is highly effective due to its ability to completely disable the model's learning capacity, it is relatively easy to detect because of its extreme uniformity. To provide a more comprehensive evaluation of Byzantine robustness, we also included subtler attacks such as the Gaussian-Noise, Gradient-Scaling, and LIE attacks. These methods are more challenging to detect and better represent real-world adversarial scenarios, where attackers often employ sophisticated techniques to evade detection.

- **Label-Flipping Attack**: The training data is targeted by altering the labels of specific samples, as reported in [18, 19, 21, 25, 22, 24]. The objective is to induce a local model trained with incorrectly labeled data. For the MNIST dataset, malicious clients relabeled their data as "0", for the HAR dataset, labels were changed to "WALKING", and for the Air Quality dataset, the labels were flipped by inverting their sign.

- **Same-Value Attack**: The learning model is compromised by setting all its parameters to a single value (as described in [24]), zero in our case. This strategy nullifies the model's ability to learn and make accurate predictions. In our scenario, the attack is executed by a malicious client when sending its local model to the central server.

- **Gaussian-Noise Attack**: Gaussian noise is introduced into the model updates by adding random perturbations to the model parameters. This noise is generated by sampling from a standard Gaussian distribution, which has a mean (μ) of 0 and a standard deviation (σ) of 1. The generated noise is then added directly to the existing parameters of the model, based on the method described in [86]. This disruption prevents the model from converging properly, thus hindering the learning process and making it harder for the model to achieve accurate predictions.

- **Gradient-Scaling Attack**: Malicious clients scale their local gradients to manipulate the learning model. Each gradient element is multiplied by a random value $\lambda \in [a, 1)$, where a is a constant determining the attack's intensity. In this experiment, a is set to 0.5, ensuring the gradients remain within a controlled range while amplifying the attack's impact, as detailed in [20].

- **LIE (Little Is Enough) Attack**: This subtle attack compromises the global model by sending slightly deviated local model updates. By carefully crafting these updates based on the mean and standard deviation of benign clients' models, the attacker aims to circumvent defenses designed to reject extreme values, as reported in [87]. In our scenario, the attack is executed by malicious clients during the transmission of their local models to the central server.

4.2 Results

In this section, we evaluate the robustness of our proposed method, introduced in Section 3, under a variety of poisoning attacks. Each attack was selected to represent a distinct adversarial strategy, targeting either the training data or the learning model itself. Our approach is designed for decentralized inference, meaning that each client performs predictions and evaluations locally. However, for the purposes of visualization, comparison, and analysis, we aggregate the local results and report global metrics that reflect the overall system performance.

To ensure a fair evaluation across different learning tasks, we adopt appropriate performance metrics for both classification and regression problems. For classification, we use the F1-score, which balances precision and recall in a single value. Precision measures the proportion of correctly predicted positive instances among all predicted positives, while recall reflects the proportion of correctly predicted positives among all actual positives. Although inference occurs locally, we compute a single global F1-score for consistency and clarity. This global score is calculated by summing the true positives (TP), false positives (FP), and false negatives (FN) across all clients, and applying the standard F1-score formula as shown in Equation (3.2). This procedure corresponds to the micro-averaged F1-score, as described in [88].

For regression tasks, we evaluate performance using the Mean Absolute Error (MAE), a widely adopted metric that quantifies the average absolute difference between predicted and true values. The global MAE is obtained by collecting all local predictions and corresponding ground truth values, then applying Equation (3.3) to the combined dataset. This mirrors the micro-averaging approach used in the classification setting, ensuring consistency across tasks.

The results presented in the following subsections demonstrate that our approach is capable of mitigating the effects of various types of adversial behaviors. It successfully counters label manipulation (Label-Flipping), neutralizes extreme model corruption (Same-Value), and resists more nuanced attacks (Gaussian-Noise, Gradient-Scaling, and LIE). These findings validate the robustness and adaptability of our approach in maintaining performance and reliability in adversarial Federated Learning scenarios.

4.2.1 Our Approach vs. Single-global-model FedAvg

To evaluate the efficacy of the proposed defense mechanisms, we conducted an experimental analysis encompassing both classification and regression tasks. In this set of experiments, our approach was first compared to the FedAvg single-global-model baseline aggregation algorithm, which operates without any defense mechanism and is henceforth referred to as the "Single-global-model FedAvg" or the "defense-less" method.

Classification Tasks: In the context of classification, experiments were performed on the MNIST and HAR datasets. Figures 10, 11 and 12 present a comparison between the proposed approach and the Single-global-model FedAvg method. In all figures, it is clear that our strategy outperforms the defense-less approach.

For the MNIST dataset, Figure 10 shows a significant difference between the proposed approach and the defense-less method for the Label-Flipping and Same-Value attacks. While the defense-less method experiences a performance drop below 0.8 with more than 20% of malicious clients, the proposed approach maintains an F1-score above 0.8 even with 90% of malicious clients. For the Gaussian-Noise attack, the defense-less method shows a performance decay to approximately 0.6 from the start, with only 10% of malicious clients, whereas the proposed approach keeps an F1-score of 0.9 even with up to 70% of malicious clients. For the Gradient-Scaling attack, the defense-less method manages to maintain an F1-score above 0.8 with up to 50% of malicious clients but is outperformed by the proposed approach, which sustains an F1-score above 0.9 even with 90% of malicious clients. Finally, under the LIE attack, the proposed strategy consistently maintains an F1-score above 0.9, even when facing 5 to 10 additional malicious clients compared to the baseline defenses.

For the HAR dataset, Figure 11 demonstrates that for the Label-Flipping attack, our approach achieves the same metrics as the defense-less method but with approximately 20% to 33% more malicious clients. For the Same-Value attack, the defense-less method quickly deteriorates, maintaining an F1-score above 0.8 only for 10% of malicious clients. In contrast, our approach maintains an F1-score above 0.8 with up to 50% malicious clients, declining slowly thereafter. For the Gaussian-Noise attack, we observe that our approach outperforms the defense-less method, maintaining the same metrics but with 20% to 55% more malicious clients. Similarly, for the Gradient-Scaling attack, our method maintains the same metrics but with 10% to 53% more malicious clients. At last, under the LIE attack, the proposed approach consistently maintains an F1-score above 0.9 even in the presence of 10 to 15 malicious clients, whereas the defense-less method manages to stay above 0.8 only when facing two or fewer malicious participants.

In this work, there are three important variables, n (number of clients), N (number of groups), and k (number of clients per group), whose values vary throughout the study.

Next, we will analyze the impact of each variable. In Figures 10 and 11, we can observe the impact of varying k. Figure 10, which corresponds to the MNIST dataset, shows the results for $k = 3$ and $k = 5$ (values chosen based on those commonly used in the literature) under the Label-Flipping and Same-Value attacks. In these results, the variation of k was barely noticeable. Similarly, Figure 11, corresponding to the HAR dataset, demonstrates a greater variation in the results, where a smaller value of k ($k = 3$) achieved superior performance. The improved results for a smaller k can be attributed to two factors: the lower number of clients per group reduces the likelihood of a malicious client compromising that group, and the HAR dataset is more heterogeneous than the MNIST dataset, which means that malicious clients can have a greater impact.

Regarding the impact of varying the number of groups (N), we can conclude that there is no significant difference in mitigation capacity across all attack types for both MNIST and HAR datasets. This trend is evident in Figures 10 and 11, where performance consistently starts to decline at the same point regardless of the value of N. However, for the HAR dataset under the Gaussian-Noise attack, it is possible to observe a difference in performance between $N = 9$ and $N = 15$ or $N = 21$, which appears to be an exception to this general trend. Overall, this observation indicates that we can opt for the smallest N, as it reduces the number of groups and the associated global models, thereby lowering the computational cost without compromising performance.

In Figures 10a, and 12, we utilized the same dataset (MNIST) and the same number of groups ($N = 15$) while varying the number of clients. Figure 10 correspond to $n = 50$, whereas Figure 12 represents $n = 30$. This allows us to observe the impact of changing the number of clients on the results. We can observe a slight difference in mitigation capacity. For 30 clients, depending on the attack, performance started to decline earlier.

Aside from the numerical results, we can highlight the architectural aspects that explain the superior performance of the proposed approach over the defense-less method. First, the use of multiple global models assigned to different groups reduces reliance on a single centralized aggregation, thereby mitigating the impact of compromised data or malicious client behavior. This decentralized structure dilutes the influence of poisoning attacks across several aggregators, making it more difficult for adversarial contributions to dominate the final model. Additionally, grouping clients leverages the natural variability of local data, which proves particularly beneficial in heterogeneous scenarios, such as the HAR dataset.

Regression Tasks: To assess the performance of our defense strategy in a regression task, we applied it to the Air Quality dataset and compared it against the Single-global-model FedAvg method under multiple poisoning attacks. Figure 13 summarizes the results for various values of N and k.

For the Label-Flipping and Gradient-Scaling attacks, the proposed strategy and

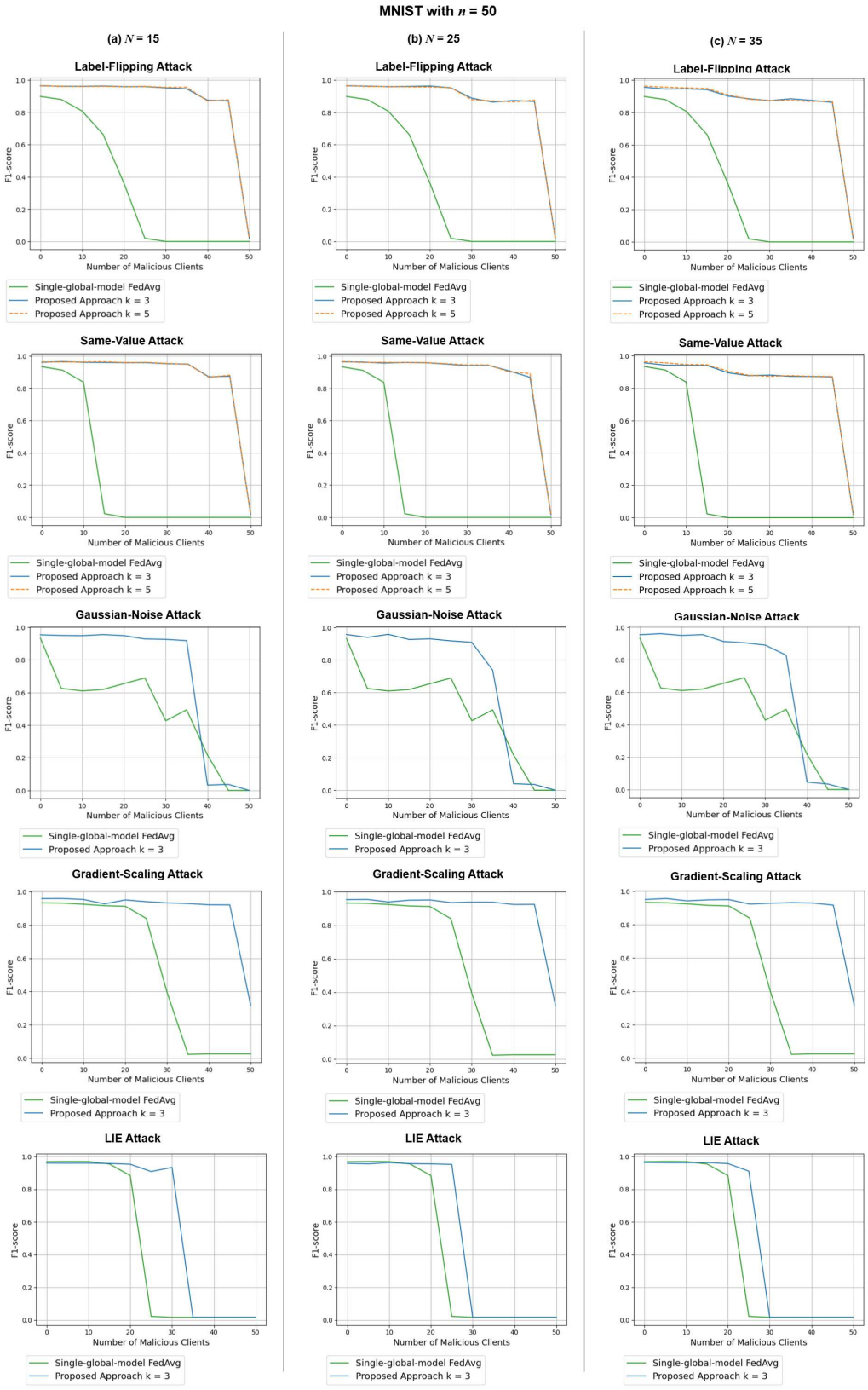

Figure 10 – Comparison of the Single-global-model FedAvg approach with the proposed approach using the MNIST dataset, with $n = 50$, N varying between 15, 25, and 35, and k varying between 3 and 5, for Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling and LIE attacks.

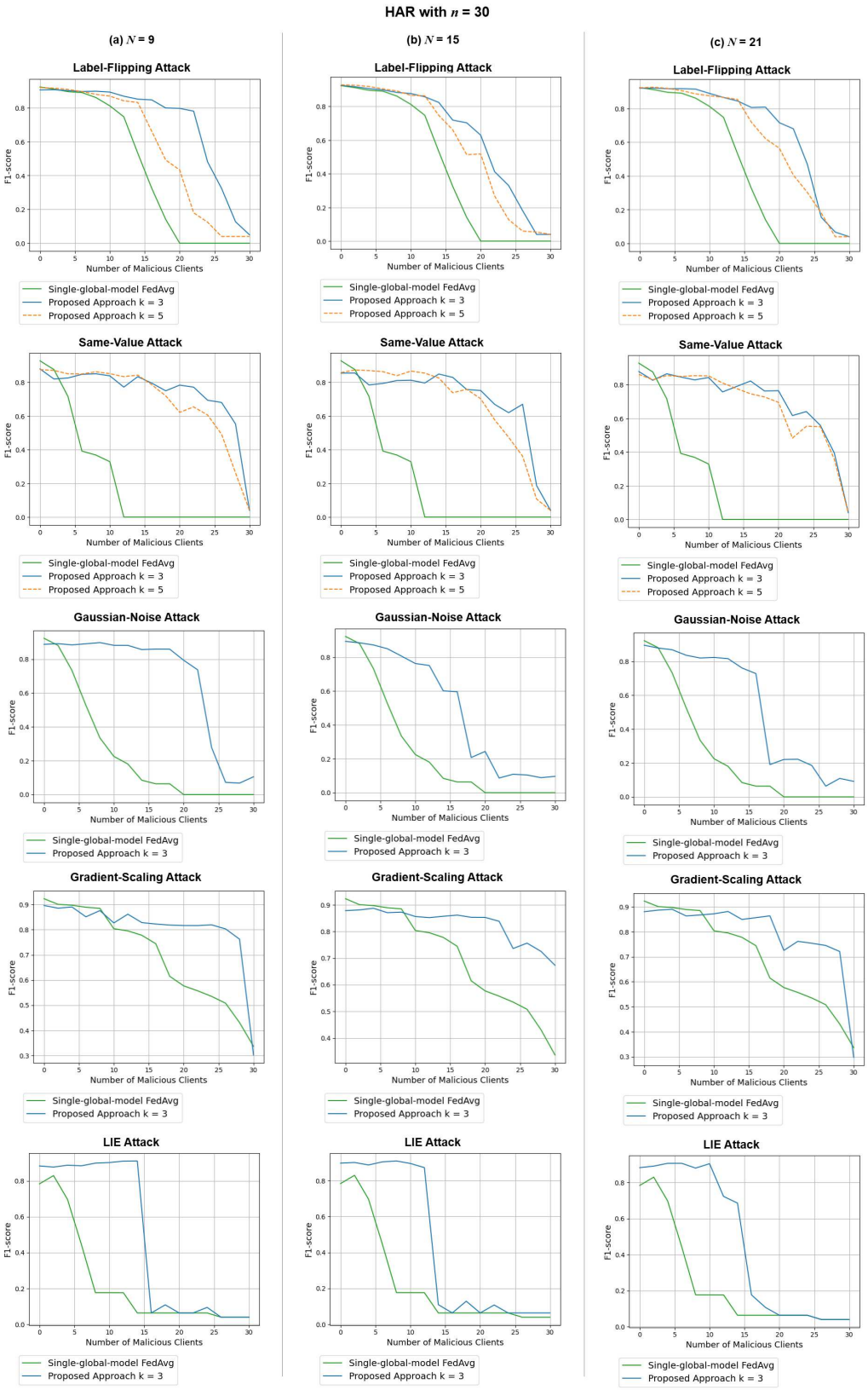

Figure 11 – Comparison of the Single-global-model FedAvg approach with the proposed approach using the HAR dataset, with $n = 30$, N varying between 9, 15, and 21, and k varying between 3 and 5, for Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling and LIE attacks.

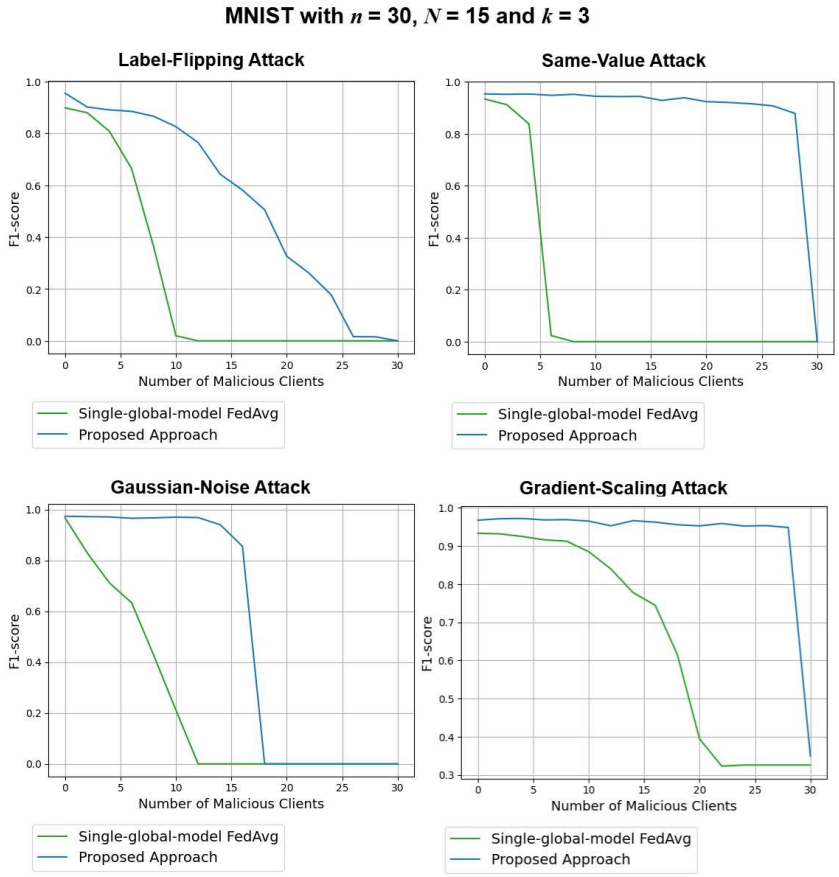

Figure 12 – Comparison of the Single-global-model FedAvg approach with the proposed approach using the MNIST dataset, with $n = 30$, $N = 15$ and $k = 3$, for Label-Flipping, Same-Value, Gaussian-Noise and Gradient-Scaling attacks.

the defense-less approach exhibit similar overall performance. Across all configurations, the results consistently show alternating outcomes, there are moments when the proposed method performs better and others when it does not. This consistent fluctuation across the graphs suggests that the proposed method does not offer a clear advantage under these specific attack types in regression tasks.

In contrast, under Same-Value and Gaussian-Noise attacks, the proposed approach consistently outperforms the Single-global-model FedAvg baseline, particularly as the number of malicious clients increases. The improvement is especially evident in configurations with $k = 2$ and $N = 5$ or $N = 8$, where the MAE remains systematically lower.

An interesting and consistent behavior is observed under the LIE attack. All approaches, including the defense-less method and the proposed approach, demonstrate strong resilience overall, with low MAE values even in the presence of malicious clients. A small spike in MAE, reaching approximately 0.2, occurs when the number of malicious clients is exactly one. This spike is seen in both methods for $N = 3$ (with $k = 3$ and $k = 4$) and, to a lesser extent, for $N = 5$ when $k = 3$. However, for $N = 8$, the increase in

MAE is observed only in the defense-less approach, while the proposed method maintains stable performance. This temporary increase in prediction error, observed when a single client executes the LIE attack, can be attributed to the nature of the attack itself. The LIE strategy introduces small, carefully crafted deviations into the local model updates, calibrated using the mean and standard deviation of the parameters from benign clients. However, with only one malicious client, the estimation of these statistics becomes unreliable. In particular, the standard deviation is undefined when computed from a single value, as it involves division by $m - 1$, where m is the number of malicious clients. When $m = 1$, this leads to division by zero, resulting in a `NaN` (Not a Number) value. If this `NaN` is used to generate the malicious update, it can contaminates the model's parameters, producing a local model with `NaN` weights. Consequently, the global model becomes unstable during the update phase, leading to a sharp degradation in performance. This behavior is a known limitation of the LIE attack, as more malicious clients are introduced, the MAE quickly returns to near-zero levels.

Regarding the regression task with the Air Quality dataset, we observe a significant variation in performance across different values of k for all attack types except LIE. However, there is no consistent pattern indicating which value of k yields the best results, as the outcomes depend on the specific attack, the number of groups (N), and the number of malicious clients. When analyzing the effect of N, for Label-Flipping and Gradient-Scaling attacks with $k = 2$, performance tends to improve as N increases. Outside of these cases, variations in N have limited impact on model robustness.

Overall, the experimental results reveal distinct patterns in the behavior of the proposed approach under different types of poisoning attacks. In the case of Label-Flipping, Gradient-Scaling, and LIE attacks, the proposed method and the Single-global-model FedAvg approach exhibit similar overall performance, with particularly close results under the LIE attack. In contrast, under Same-Value and Gaussian-Noise attacks, the proposed method consistently outperforms the defense-less baseline, especially as the number of malicious clients increases.

4.2.2 Separated Defenses

As previously mentioned, our approach combines three defense strategies. In this subsection, we assess the individual effectiveness of each strategy to better understand their contributions to the overall defense mechanism. By testing separate configurations, we aim to highlight that although each strategy can offer some level of resilience against poisoning attacks, they are significantly more effective when combined.

We examine two feasible combinations: one that pairs group division with the evaluation of global model performance, and another that combines group division with a voting-based inference method. These are the only viable configurations for individual

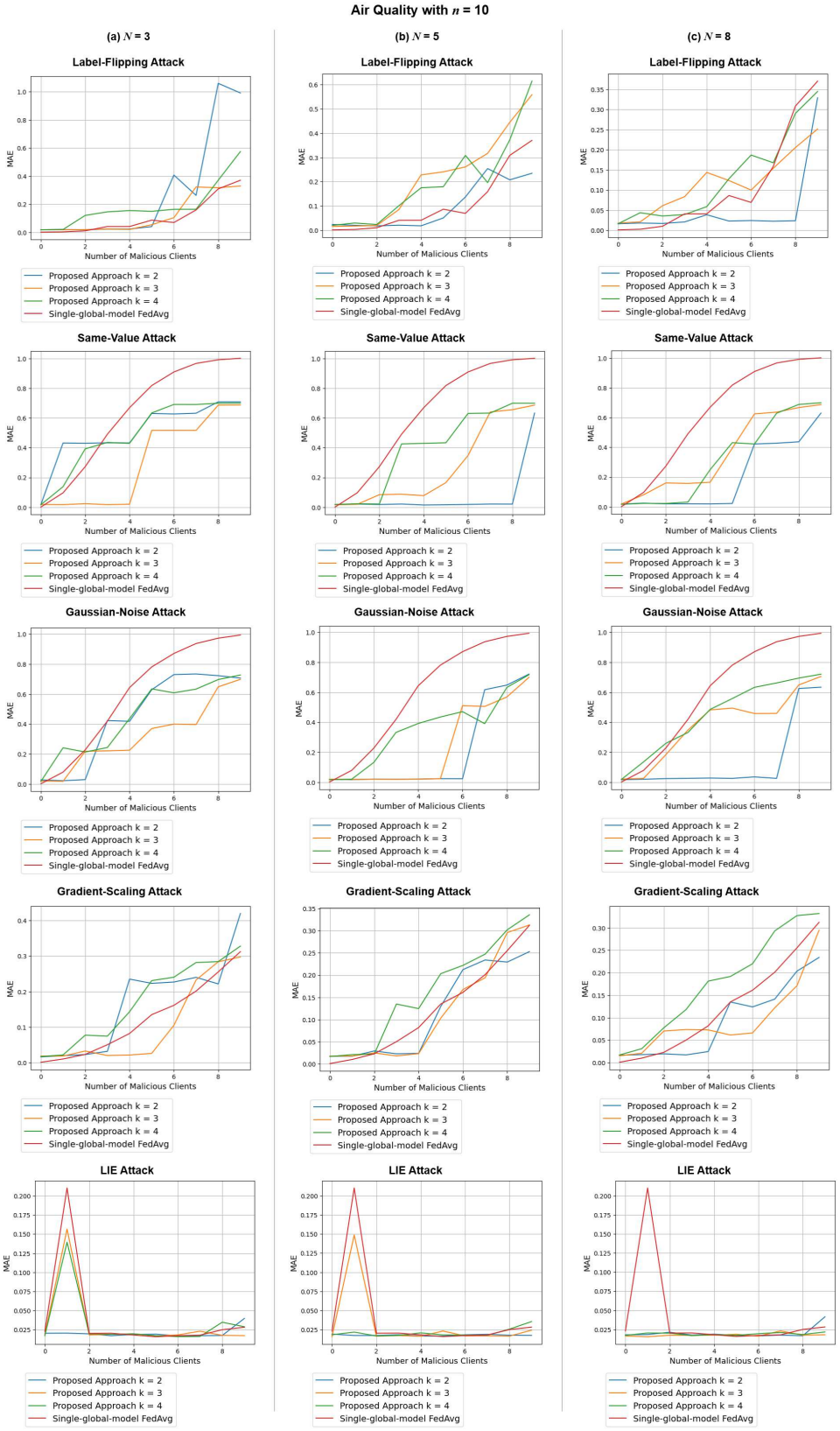

Figure 13 – Comparison of the Single-global-model FedAvg approach with the proposed approach using the Air Quality dataset, with $n = 10$, N varying between 3, 5, and 8, and k varying between 2, 3 and 4, for Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling and LIE attacks.

assessment, as both the evaluation mechanism and the voting process depend on the existence of client groups to operate effectively. Without group division, their logic cannot be meaningfully applied. At the same time, group division alone does not provide robust defense against adversarial clients, as it lacks mechanisms to detect or mitigate malicious behavior.

Classification Tasks: In the context of classification, for the first variation (defenses 1 and 2), the final voting step is excluded. The n clients are first divided into N groups, with each group consisting of k clients. Once the groups are defined, local training begins. Afterward, the clients send their local models to the central server, which aggregates them based on the predefined groups. Following this aggregation, clients receive all the aggregated global models and evaluate their performance using private validation datasets. Each client selects the global model with the highest F1-score as their new local model. This process is repeated iteratively until the training concludes. During inference, clients receive all global models, select the one with the best validation performance as their local model, and use it to make predictions on their local data.

The second variation (defenses 1 and 3) follows a similar initial process. The n clients are first divided into N groups, with each group consisting of k clients, and local training is initiated. After training, clients send their local models to the central server, which aggregates the models based on the predefined groups. Each client then receives the global model from the group they belong to (or randomly selected from the groups they participate in, if they are part of multiple groups). This iterative process continues until the training is complete. Once training finishes, the inference phase begins, relying on the previously described voting method.

In Figure 14, we present the results obtained using the MNIST dataset. Across all tested attacks, the proposed approach outperformed methods that rely on only two out of the three defenses. Notably, for three of these attacks (Label-Flipping, Same-Value, and Gradient-Scaling), the F1-score remained above 0.82 even with 90% of the clients being malicious. For the other tested attacks (LIE and Gaussian-Noise), the F1-score exceeded 0.85 even with 60% to 70% of the clients being malicious.

In Figure 15, we show the results for the HAR dataset. Similar to MNIST, the proposed approach consistently outperformed techniques that utilize only two of the three defenses across all attacks. Particularly noteworthy is the Gradient-Scaling attack, where the F1-score remained above 0.8 even with 90% of the clients being malicious.

A deeper analysis shows that combining the three defense strategies provides stronger protection against poisoning attacks. Group division alone is insufficient, as it only separates client data without addressing adversarial behavior. The global model performance evaluation (defense 2) serves as a filter, enabling clients to choose the best-performing model and enhancing the final model's robustness. The voting mechanism

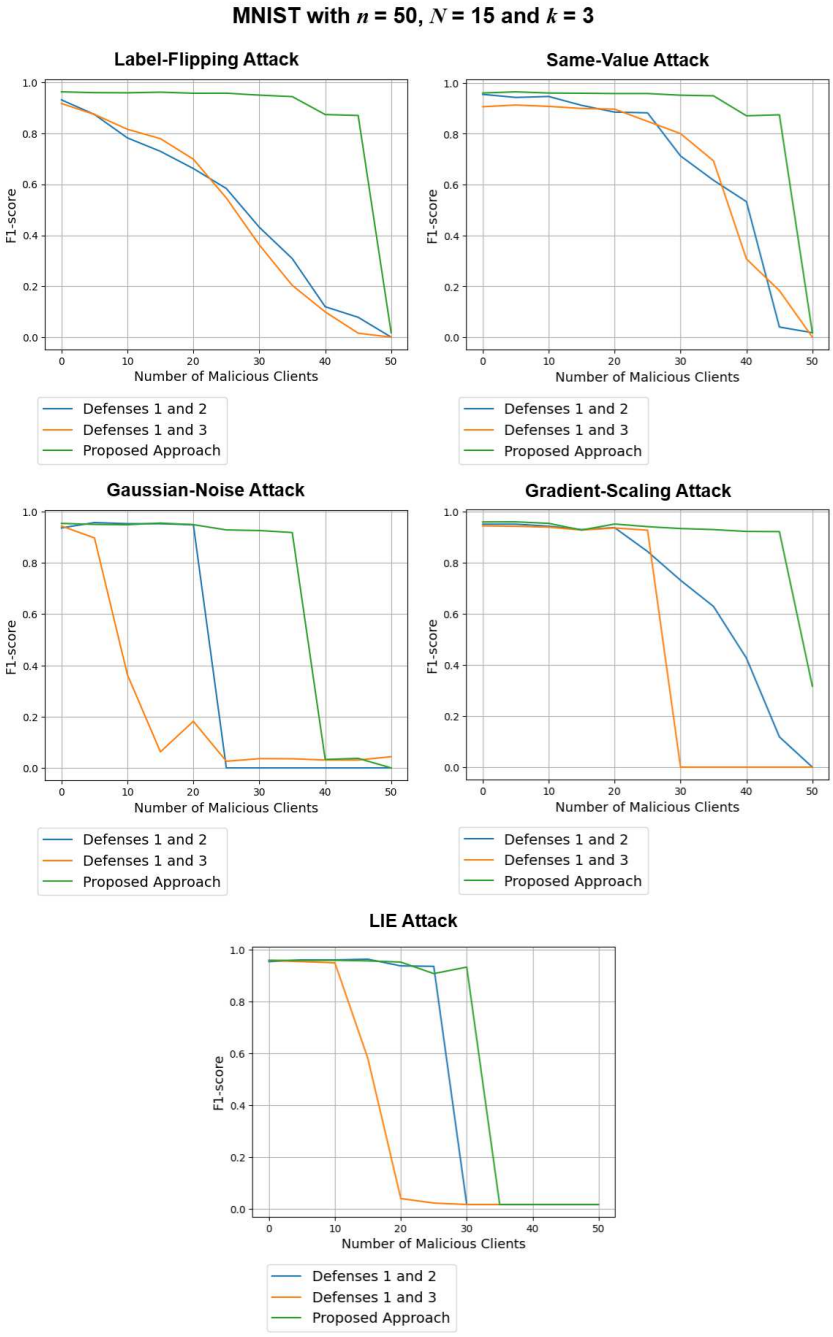

Figure 14 – Comparison of the proposed approach with two variations, each isolating different defense combinations, using the MNIST dataset with $n = 50$, $N = 15$, and $k = 3$, for Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling and LIE attacks. This comparison evaluates the performance of Defenses 1 and 2, and Defenses 1 and 3, separately.

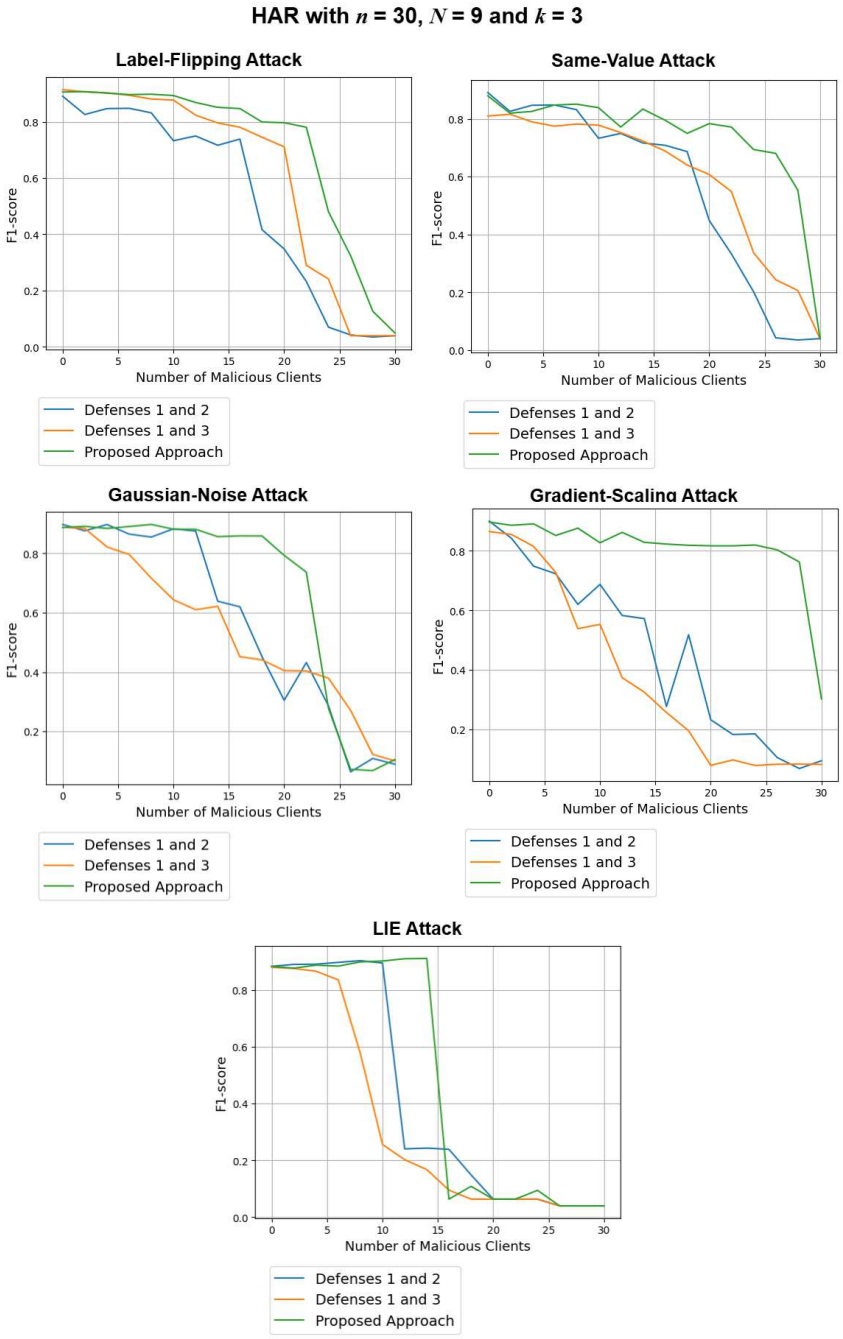

Figure 15 – Comparison of the proposed approach with two variations, each isolating different defense combinations, using the HAR dataset with $n = 30$, $N = 9$, and $k = 3$, for Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling and LIE attacks. This comparison evaluates the performance of Defenses 1 and 2, and Defenses 1 and 3, separately.

(defense 3) further strengthens the defense by enabling collective decision-making, minimizing the impact of adversarial data. In conclusion, the integration of these strategies creates a robust and adaptable defense system, significantly improving resilience against various poisoning attacks.

Regression Tasks: In regression tasks, for the first variation (defenses 1 and 2), the final weighted inference step is excluded. The n clients are first divided into N groups, with each group consisting of k clients. Once the groups are defined, local training begins. Afterward, the clients send their local models to the central server, which aggregates them based on the predefined groups. Following this aggregation, clients receive all the aggregated global models and evaluate their performance using private validation datasets. Each client selects the global model with the MAE as their new local model. This process is repeated iteratively until the training concludes. Then, during inference phase, clients receive all global models, select the one with the best validation performance as their local model, and use it to make predictions on their local data.

The second variation (defenses 1 and 3) begins with a process similar to the first. The n clients are divided into N groups, each containing k clients, and local training is performed within these groups. After training, the clients send their local models to the central server, which aggregates them according to their predefined groupings. Each client then receives the global model corresponding to its group, or, if the client belongs to multiple groups, a global model randomly selected from among them. During inference, the clients apply a weighted average over the predictions of the global models. The weights are computed from the MAE scores obtained during the evaluation step, where models with lower error receive higher weight. Given a test input x, each global model produces a prediction, and the final output is calculated as a weighted average of the predictions.

Figure 16 presents the results for the Air Quality dataset under various attack scenarios. Across all experiments, a consistent trend emerges: the combination of defenses 1 and 3 systematically underperforms compared to both the full defense strategy and the pairwise combination of defenses 1 and 2. This observation suggests that the third defense component, responsible for the weighted average inference, does not contribute positively to robustness in regression tasks; on the contrary, its inclusion appears to degrade overall performance.

Moreover, in most attack settings, the combination of defenses 1 and 2 not only outperforms the variant including defenses 1 and 3 but also exceeds the performance of the proposed approach. This pattern holds across multiple configurations and attack types, except under the Gaussian-Noise attack, where the proposed approach achieves the best performance when the number of malicious clients exceeds three. Additionally, in the case of the LIE attack, the proposed approach and the combination of defenses 1 and 2 demonstrate very similar performance. These results suggest that, instead of

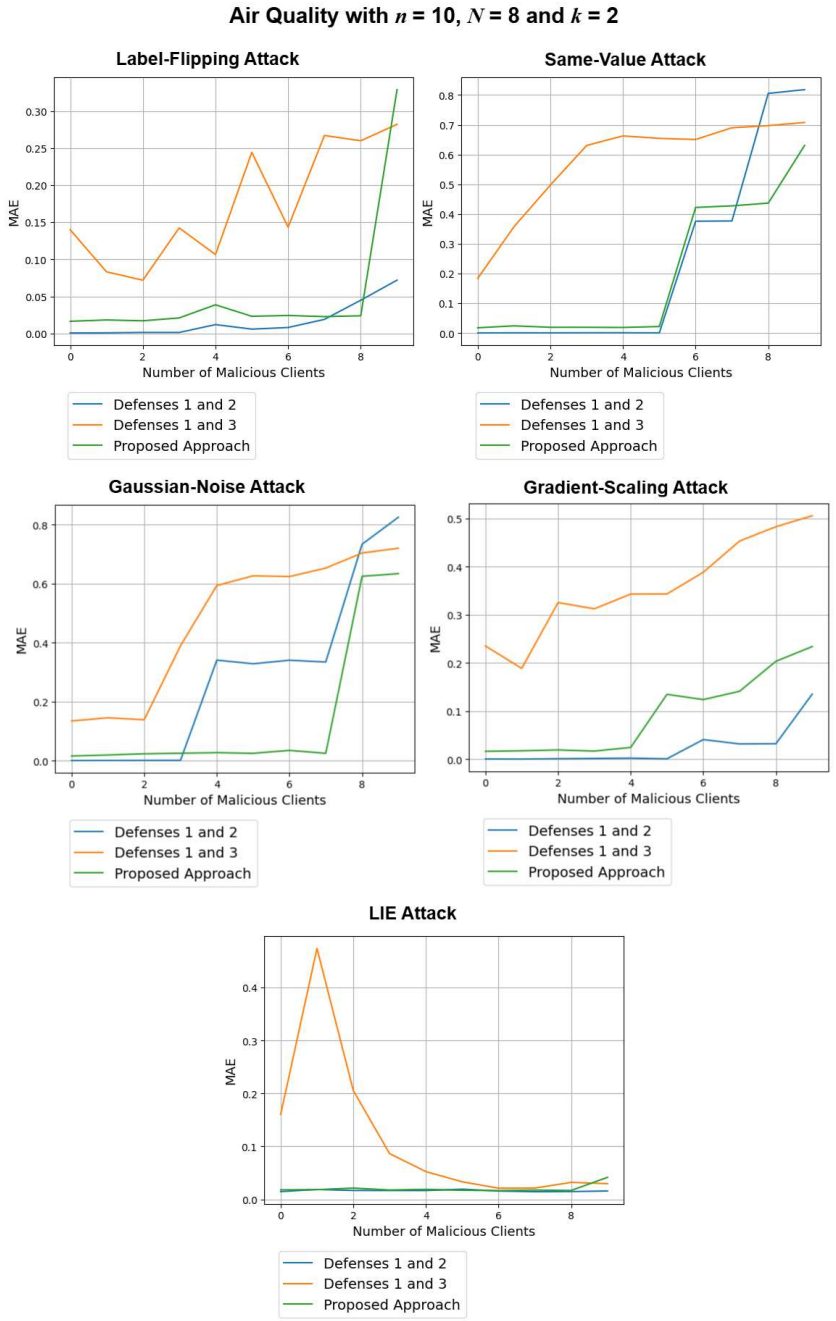

Figure 16 – Comparison of the proposed approach with two variations, each isolating different defense combinations, using the Air Quality dataset with $n = 10$, $N = 8$, and $k = 2$, for Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling and LIE attacks. This comparison evaluates the performance of Defenses 1 and 2, and Defenses 1 and 3, separately.

enhancing resilience as intended, the third defense step may actually reduce effectiveness in regression tasks by introducing instability and compromising robustness.

4.2.3 Our Approach vs. Other Existing Defenses

In this subsection, we compare the proposed defense strategy with well-established defense methods, namely Krum, Trimmed Mean, and Median. These methods represent some of the most widely used techniques in Federated Learning to address adversarial threats. By analyzing the performance of these methods across different experimental conditions, we can gain insights into the relative strengths and limitations of each defense technique. Furthermore, this comparison helps to highlight the unique advantages of our approach, especially in scenarios where traditional defenses may fall short.

Classification Tasks: In Figure 17, we present the results obtained using the MNIST dataset. Across all tested attacks, the proposed approach outperformed other existing defenses. For the Label-Flipping and Same-Value attacks, there is a significant difference, where our approach maintains an F1-score above 0.8 even with 90% of the clients being malicious, while other approaches achieve the same F1-score for at most 20% of malicious clients. On the other hand, the Gaussian-Noise and Gradient-Scaling attacks are more subtle, leading to defense strategies producing more similar results, with our approach performing marginally better.

In Figure 18, we show the results for the HAR dataset. Similar to MNIST, the proposed approach performed better than other existing defenses. For the Label-Flipping, Gaussian-Noise, and Gradient-Scaling attacks, our approach achieved an F1-score above 0.8, even with approximately 20% more malicious clients compared to other defenses. For the Same-Value attack, our approach maintained an F1-score above 0.7 even with 90% of the clients being malicious, while other defense strategies achieved the same F1-score with only about 20% of malicious clients. In the case of the LIE attack, the proposed approach also demonstrates strong resilience, maintaining an F1-score above 0.95 even with 60% of the clients being malicious. In contrast, existing defenses sustain this level of performance only when up to 30% of the clients are adversarial.

While methods like Krum, Trimmed Mean, and Median rely primarily on model aggregation, our approach benefits from group division, global model evaluation, and voting, which help dilute the influence of malicious clients. This results in superior performance, especially for attacks like Label-Flipping and Same-Value, where our approach maintains high F1-scores even with a large percentage of malicious clients.

Regression Tasks: Figure 19 presents the results obtained using the Air Quality dataset across various poisoning attack scenarios. Overall, the proposed approach exhibits performance comparable to other defense strategies for the Label-Flipping, Gradient-Scaling, and LIE attacks. Notably, the Krum defense shows sharp spikes in MAE under the Label Flipping and LIE attacks, specifically when there are seven and one malicious clients, respectively. Meanwhile, the proposed method maintains stable and consistent

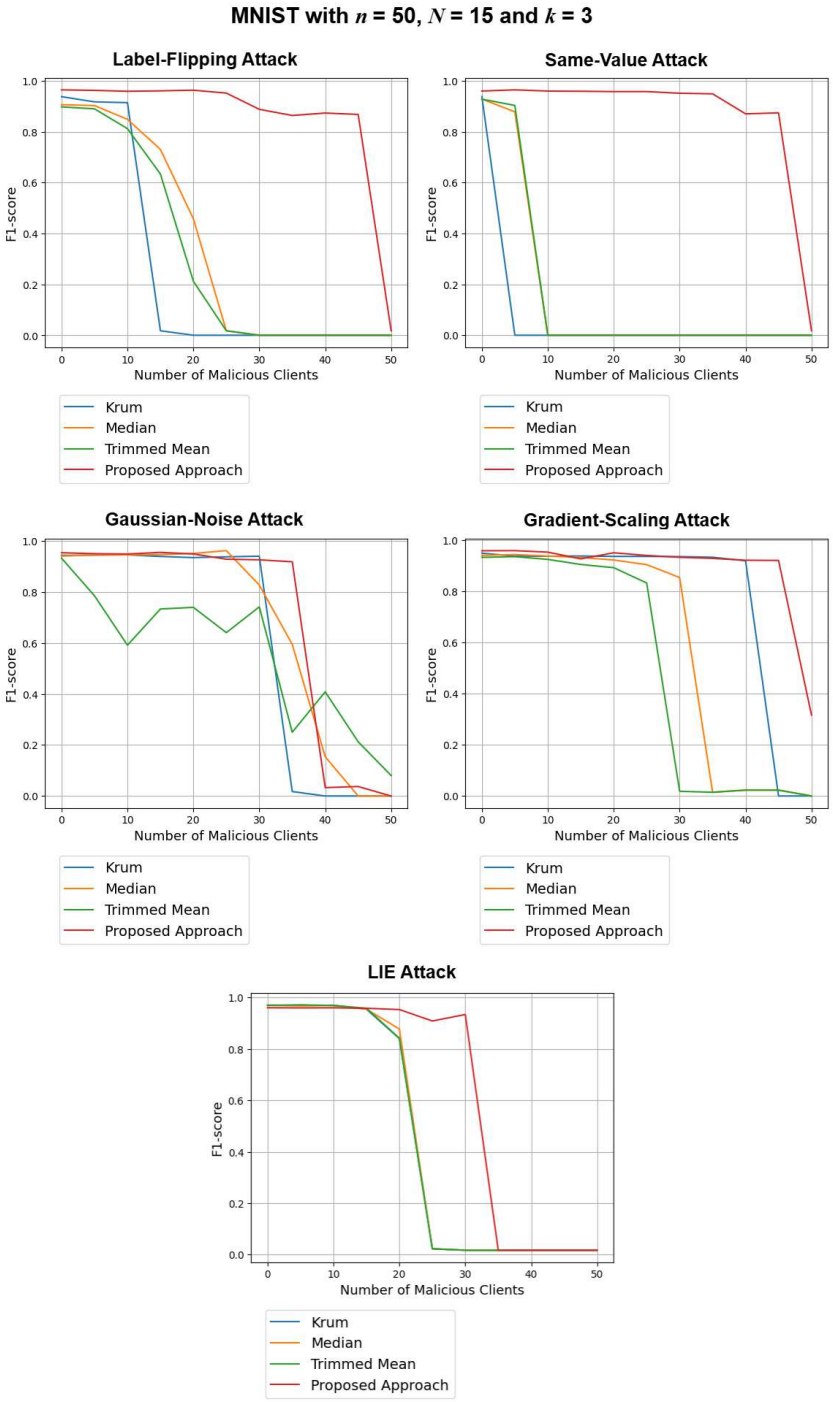

Figure 17 – Comparison of the proposed approach with Krum, Trimmed Mean, and Median, using the MNIST dataset with $n = 50$, $N = 15$, and $k = 3$, for Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling and LIE attacks.

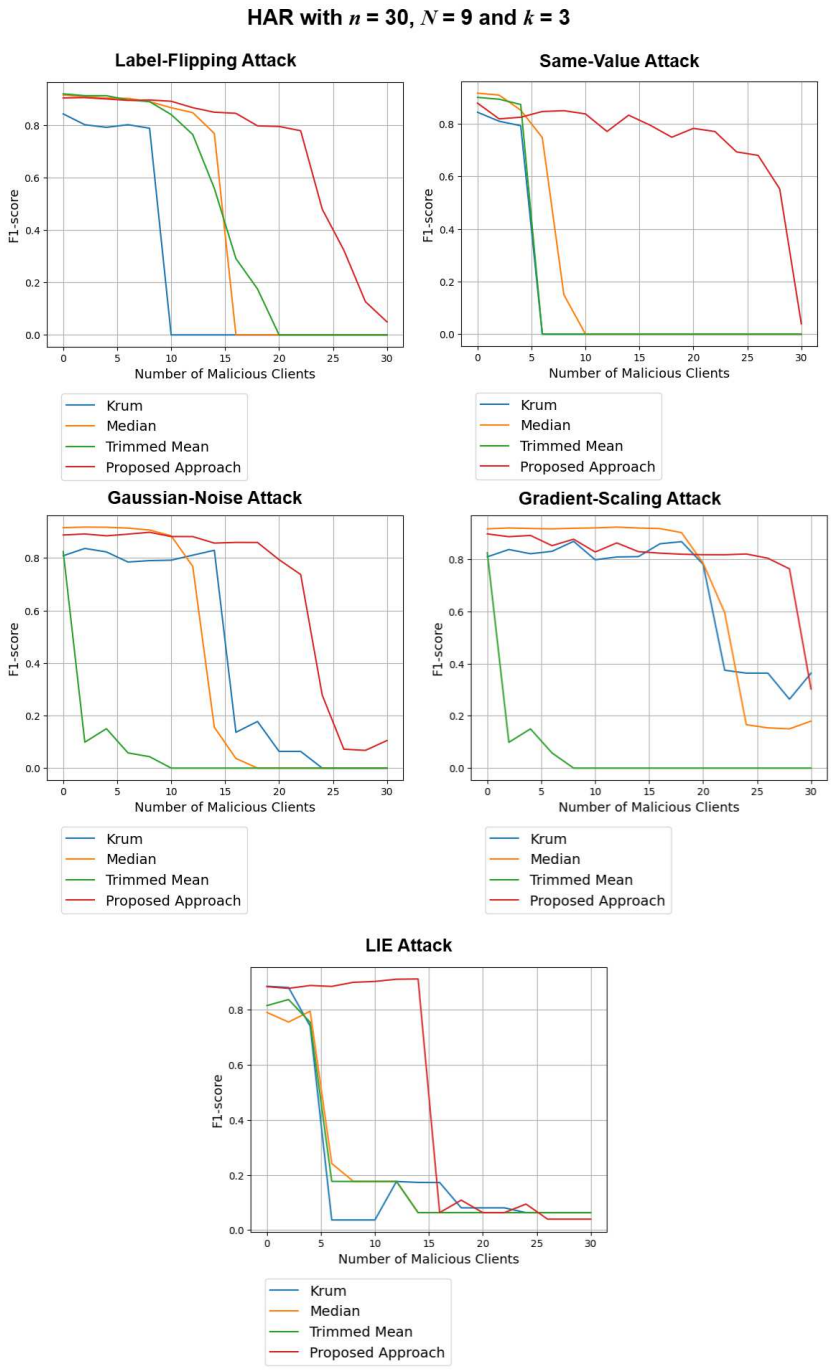

Figure 18 – Comparison of the proposed approach with Krum, Trimmed Mean, and Median, using the HAR dataset with $n = 30$, $N = 9$, and $k = 3$, for Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling and LIE attacks.

MAE values, comparable to those of the other defenses.

Regarding the Same-Value attack, the proposed approach demonstrates superior robustness compared to all other methods. It maintains an MAE value close to zero for up to 50% of malicious clients. Beyond this threshold, the increase in MAE is more gradual, whereas competing defenses exhibit a much more abrupt degradation.

For the Gaussian-Noise attack, the proposed strategy outperforms both the Median and Trimmed Mean defenses, maintaining MAE values near zero for up to seven malicious clients. Although Krum performs slightly better by sustaining low MAE up to eight malicious clients, its performance degrades more sharply thereafter.

These findings highlight the overall effectiveness of the proposed approach by maintaining stable MAE values and outperforming or matching existing defenses across most attacks, the method demonstrates its potential as a robust solution for Federated Learning environments vulnerable to poisoning threats in regression tasks.

4.2.4 Dealing with Malicious Selections

As mentioned in Subsection 3.1, during the performance evaluation phase of our method, a malicious client may attempt to compromise the system by selecting the worst-performing global model, i.e., the one with the lowest F1-score, instead of the best, in an effort to degrade overall system performance. In Figure 20, the curve labeled "Malicious Selections" represents exactly this adversarial scenario: our proposed method is still being used, but the malicious clients intentionally misreport their evaluations to favor the worst model. Despite this, we observe that our approach remains superior to the Single-global-model FedAvg approach throughout the experiment. Furthermore, the results indicate that model performance remains stable even with malicious votes, up until approximately 50% of the clients are malicious.

4.3 Discussion of Results

In this work, we evaluated the proposed method on two types of tasks: classification and regression. Experiments were conducted on three different datasets: MNIST and HAR for classification, and Air Quality for regression. We also tested the method against five different attack types: Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling, and LIE. For classification tasks, the F1-score metric was used to assess performance, while MAE was employed for regression tasks.

Regarding the classification tasks, when comparing the proposed approach with a baseline without defense, the proposed method outperformed the defense-less method in all cases. In the majority of these cases, the improvement was significant, both for the MNIST and HAR datasets, showing consistent and reliable results. Additionally, when

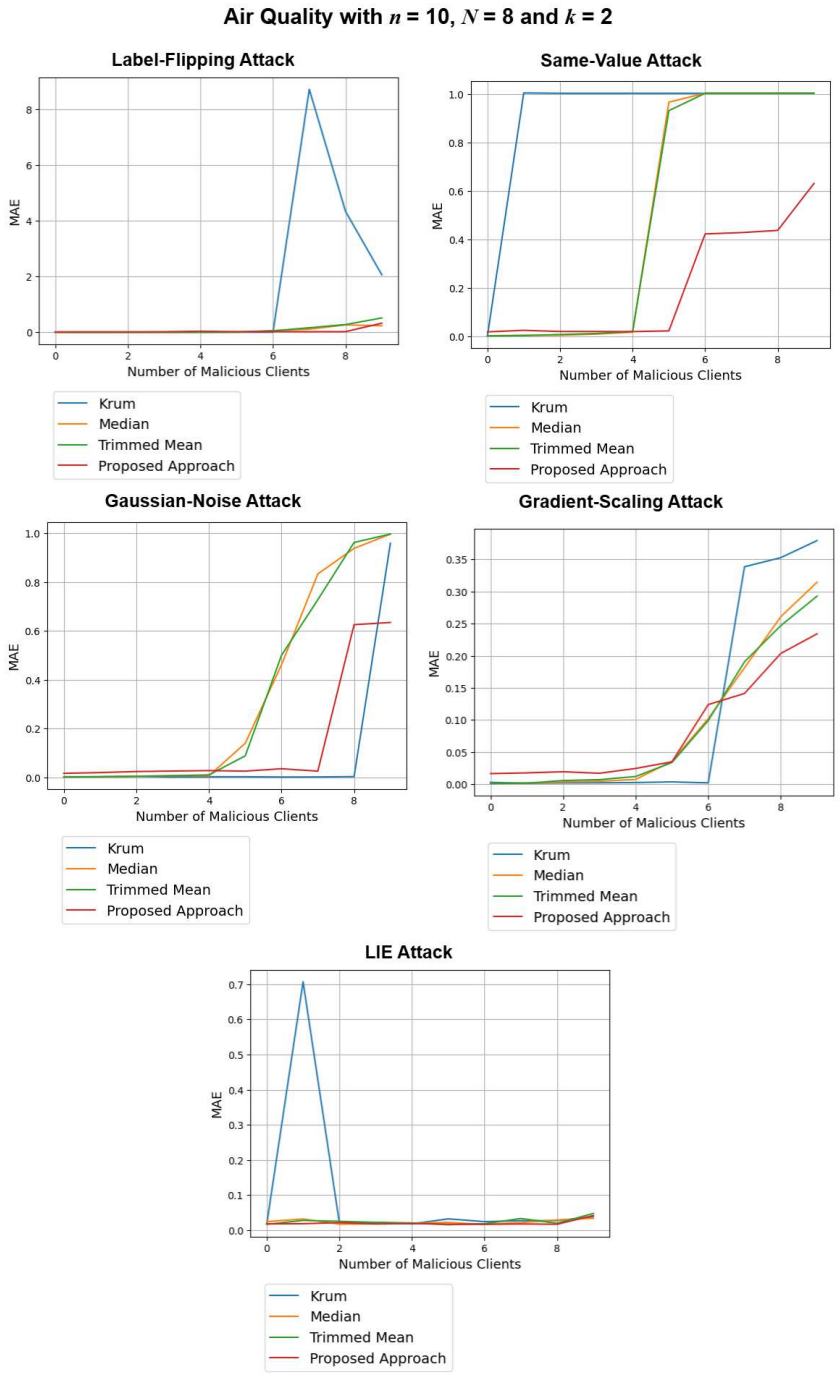

Figure 19 – Comparison of the proposed approach with Krum, Trimmed Mean, and Median, using the Air Quality dataset with $n = 10$, $N = 8$, and $k = 2$, for Label-Flipping, Same-Value, Gaussian-Noise, Gradient-Scaling and LIE attacks.

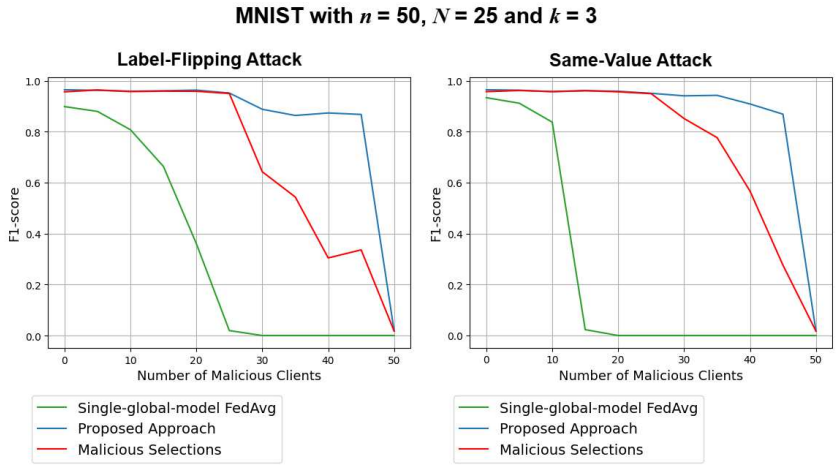

Figure 20 – Analysis of the impact of clients issuing malicious votes using the MNIST dataset with $n = 50$, $N = 25$ and $k = 3$.

evaluating the defense components of the proposed approach separately, we observed that the combined strategy consistently surpassed the isolated defenses, further demonstrating the potential and synergy of integrating the three defense strategies, again for both MNIST and HAR. Comparing with other existing defense methods, the proposed approach either outperformed or achieved comparable results in almost all cases, frequently by a considerable margin.

For regression tasks, the proposed method outperformed the baseline without defense under Same-Value and Gaussian-Noise attacks, while achieving comparable results in the presence of Label-Flipping, Gradient-Scaling, and LIE attacks. Against other existing defense strategies, our approach showed either superior or on-par results across the evaluated attacks. Notably, when evaluating the defense components of the proposed approach separately, an analysis of the individual defense techniques revealed that the third method, which relies on weighted averaging, was less effective in regression contexts and, at times, even detrimental to the model's robustness. Contrary to its intended purpose, the third defense step appears to hinder overall effectiveness by introducing instability or disrupting the learning dynamics, ultimately compromising the model's resilience.

These findings suggest that while the overall structure of the proposed method shows promise across both classification and regression tasks, specific adjustments are necessary to improve its performance in regression scenarios. Future work should focus on refining the third defense strategy for regression tasks, possibly exploring alternative techniques more suitable for continuous output spaces.

4.4 Limitations

Although the proposed method enhances system robustness and resilience to adversarial behavior, certain limitations must be acknowledged. The approach integrates three distinct defense strategies which, although effective, contribute to increased system complexity. This added complexity results in higher computational demands and structural overhead, potentially impacting scalability. Moreover, the computational, communication, and storage costs associated with the proposed method have not been formally quantified.

Despite its robust defense mechanisms, one key limitation lies in its inability to distinguish between model degradation caused by data quality issues, such as poor, noisy, or imbalanced data, and degradation resulting from malicious behavior. Consequently, during the evaluation phase, global models affected by benign data issues may be mistakenly discarded under the assumption of adversarial corruption. This can reduce the overall diversity and representativeness of the aggregated model, especially in scenarios where data quality is heterogeneously distributed among clients.

Moreover, the system's effectiveness relies on the assumption that the majority of participating global models are trustworthy. In situations where a significant number of malicious clients are strategically distributed across different client groups, there is a risk that multiple global models may be simultaneously compromised. This coordinated interference can diminish the method's ability to effectively filter out poisoned models during the voting phase, thereby reducing its defense efficacy in more extreme attack scenarios or highly imbalanced settings.

Another limitation arises when analyzing the performance of individual defense components, particularly in regression tasks. Experimental results indicate that the third defense mechanism, based on weighted averaging, performs suboptimally in regression contexts. Rather than enhancing robustness, this component sometimes introduces instability and negatively impacts the learning dynamics.

Finally, the current evaluation has been restricted to untargeted poisoning attacks. Targeted attacks, such as backdoor attacks, have not been explored within this work and remain an open avenue for future investigation.

5 CONCLUSION

Federated Learning has emerged as a transformative approach in the field of Machine Learning, addressing critical challenges in data privacy and scalability. By enabling the decentralized training of models, FL offers significant advantages in preserving user confidentiality and supporting the development of Machine Learning applications in privacy-sensitive domains such as healthcare, finance, and IoT systems. However, the decentralized nature of FL also introduces unique vulnerabilities, particularly to poisoning attacks, which can severely degrade the performance and reliability of global models. These attacks, whether through data or model poisoning, highlight the pressing need for robust defense mechanisms to ensure the integrity and trustworthiness of FL systems.

In this work, an FL defense system is proposed, combining three different techniques to mitigate poisoning attacks. The approach divides the clients into randomly sampled groups, evaluates the global models performance using the client's private datasets, and, during its final step, uses a voting scheme to predict the labels. We assessed the effectiveness of our proposal across three datasets and two types of tasks (classification and regression) against five different attacks, four targeting the model and one targeting the data. This comprehensive evaluation of Byzantine robustness makes detection more challenging and better reflects real-world scenarios.

The proposed solution demonstrated strong robustness across all evaluated classification scenarios, consistently outperforming both a baseline Federated Learning setup without defenses and established defense mechanisms. In regression scenarios, the proposed approach achieved comparable performance or exceeded the baseline and other existing defense strategies. However, the third defense component, based on weighted averaging, showed limited effectiveness in continuous output spaces, negatively affecting model stability. These results indicate that the current formulation of this defense step may not generalize well to regression contexts.

Future work will aim to advance the mitigation of poisoning attacks in regression tasks, a scenario that continues to pose considerable challenges with respect to robustness and predictive performance. Particular attention will be devoted to enhancing the proposed method, with the objective of improving its effectiveness in regression contexts. In this regard, it will be necessary to explore alternative defense mechanisms to the weighted averaging strategy, which demonstrated limited efficacy and, in some instances, negatively impacted robustness. Additionally, future studies will seek to evaluate these refinements using new datasets, in order to further assess the generalizability and reliability of the proposed approach across diverse domains and application settings.

BIBLIOGRAPHY

[1] LIU, B. et al. When machine learning meets privacy: A survey and outlook. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 54, n. 2, mar 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3436755>.

[2] YANG, Q. et al. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, Association for Computing Machinery, New York, NY, USA, v. 10, n. 2, jan 2019. ISSN 2157-6904. Disponível em: <https://doi.org/10.1145/3298981>.

[3] MCMAHAN, B.; RAMAGE, D. *Federated Learning: Collaborative Machine Learning without Centralized Training Data*. 2017. Accessed on june 06, 2024. Disponível em: <https://research.google/blog/federated-learning-collaborative-machine-learning-without-centralized-training-data>.

[4] WITT, L. et al. Decentral and incentivized federated learning frameworks: A systematic literature review. *IEEE Internet of Things Journal*, v. 10, n. 4, p. 3642–3663, 2023.

[5] ZHANG, C. et al. A survey on federated learning. *Knowledge-Based Systems*, Elsevier, v. 216, p. 106775, 2021.

[6] SHOKRI, R. et al. Membership inference attacks against machine learning models. In: IEEE. *2017 IEEE Symposium on Security and Privacy (SP)*. [S.l.], 2017. p. 3–18.

[7] FREDRIKSON, M.; JHA, S.; RISTENPART, T. Model inversion attacks that exploit confidence information and basic countermeasures. In: ACM. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. [S.l.], 2015. p. 1322–1333.

[8] SZEGEDY, C. et al. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2014.

[9] GOODFELLOW, I. J.; SHLENS, J.; SZEGEDY, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2015.

[10] BAGDASARYAN, E. et al. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2020.

[11] FANG, M. et al. Local model poisoning attacks to byzantine-robust federated learning. *arXiv preprint arXiv:1911.11815*, 2020.

[12] FUNG, C.; YOON, C. J.; BESCHASTNIKH, I. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2020.

[13] BHAGOJI, A. N. et al. Analyzing federated learning through an adversarial lens. *International Conference on Machine Learning (ICML)*, p. 634–643, 2019.

[14] WANG, Z. et al. *Defense Strategies Toward Model Poisoning Attacks in Federated Learning: A Survey*. 2022.

[15] FANG, M. et al. Local model poisoning attacks to Byzantine-Robust federated learning. In: *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 2020. p. 1605–1622. ISBN 978-1-939133-17-5. Disponível em: <https://www.usenix.org/conference/usenixsecurity20/presentation/fang>.

[16] TOLPEGIN, V. et al. Data poisoning attacks against federated learning systems. In: CHEN, L. et al. (Ed.). *Computer Security – ESORICS 2020*. Cham: Springer International Publishing, 2020. p. 480–501. ISBN 978-3-030-58951-6.

[17] BOUACIDA, N.; MOHAPATRA, P. Vulnerabilities in federated learning. *IEEE Access*, v. 9, p. 63229–63249, 2021.

[18] XU, C. et al. Tdfl: Truth discovery based byzantine robust federated learning. *IEEE Transactions on Parallel and Distributed Systems*, v. 33, n. 12, p. 4835–4848, 2022.

[19] LI, S.; NGAI, E.; VOIGT, T. Byzantine-robust aggregation in federated learning empowered industrial iot. *IEEE Transactions on Industrial Informatics*, v. 19, n. 2, p. 1165–1175, 2023.

[20] CHE, C. et al. A decentralized federated learning framework via committee mechanism with convergence guarantee. *IEEE Transactions on Parallel and Distributed Systems*, Institute of Electrical and Electronics Engineers (IEEE), v. 33, n. 12, p. 4783–4800, dez. 2022. ISSN 2161-9883. Disponível em: <http://dx.doi.org/10.1109/TPDS.2022.3202887>.

[21] JEBREEL, N. M. et al. Enhanced security and privacy via fragmented federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, v. 35, n. 5, p. 6703–6717, 2024.

[22] ZHANG, Z. et al. Safelearning: Secure aggregation in federated learning with backdoor detectability. *IEEE Transactions on Information Forensics and Security*, Institute of Electrical and Electronics Engineers (IEEE), v. 18, p. 3289–3304, 2023. ISSN 1556-6021. Disponível em: <http://dx.doi.org/10.1109/TIFS.2023.3280032>.

[23] CAO, X.; JIA, J.; GONG, N. Z. Provably secure federated learning against malicious clients. *CoRR*, abs/2102.01854, 2021. Disponível em: <https://arxiv.org/abs/2102.01854>.

[24] CAO, X. et al. Flcert: Provably secure federated learning against poisoning attacks. *IEEE Transactions on Information Forensics and Security*, v. 17, p. 3691–3705, 2022.

[25] ANDREINA, S. et al. Baffle: Backdoor detection via feedback-based federated learning. *CoRR*, abs/2011.02167, 2020. Disponível em: <https://arxiv.org/abs/2011.02167>.

[26] MITCHELL, T. et al. Machine learning. *Annual Review of Computer Science*, v. 4, n. 1, p. 417–433, 1990. Disponível em: <https://doi.org/10.1146/annurev.cs.04.060190.002221>.

[27] JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. *Machine learning and deep learning*. 2021.

[28] KOTSIANTIS, S.; ZAHARAKIS, I.; PINTELAS. Machine learning: a review of classification and combining techniques. *Artif Intell Rev*, v. 26, n. 1, p. 159–190, 2006. Disponível em: <https://doi.org/10.1007/s10462-007-9052-3>.

[29] MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of machine learning*. [S.l.: s.n.], 2018.

[30] BISHOP, C. M.; NASRABADI, N. M. *Pattern recognition and machine learning*. [S.l.]: Springer, 2006. v. 4.

[31] SILVER, D. et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, v. 362, n. 6419, p. 1140–1144, 2018. Disponível em: <https://www.science.org/doi/abs/10.1126/science.aar6404>.

[32] JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. *CoRR*, abs/2104.05314, 2021. Disponível em: <https://arxiv.org/abs/2104.05314>.

[33] WANG, S.-C.; WANG, S.-C. Artificial neural network. *Interdisciplinary computing in java programming*, Springer, p. 81–100, 2003.

[34] MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

[35] FUKUSHIMA, K. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, Springer, v. 20, n. 3, p. 121–136, 1975.

[36] LIPPMANN, R. An introduction to computing with neural nets. *IEEE Assp magazine*, IEEE, v. 4, n. 2, p. 4–22, 1987.

[37] GURNEY, K. *An introduction to neural networks*. [S.l.]: CRC press, 2018.

[38] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

[39] HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural networks*, Elsevier, v. 2, n. 5, p. 359–366, 1989.

[40] HINTON, G. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, IEEE, v. 29, n. 6, p. 82–97, 2012.

[41] ZHU, X. et al. Deep learning for financial applications: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, 2021.

[42] KONONENKO, I. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, Elsevier, v. 23, n. 1, p. 89–109, 2001.

[43] LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

[44] VASWANI, A. et al. Attention is all you need. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. v. 30.

[45] LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, IEEE, v. 86, n. 11, p. 2278–2324, 1998.

[46] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016.

[47] NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. 2010.

[48] SCHERER, D.; MÜLLER, A.; BEHNKE, S. Evaluation of pooling operations in convolutional architectures for object recognition. In: SPRINGER. *International conference on artificial neural networks*. [S.l.], 2010. p. 92–101.

[49] SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 1, p. 1929–1958, 2014.

[50] IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: PMLR. *International conference on machine learning*. [S.l.], 2015. p. 448–456.

[51] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. v. 25, p. 1097–1105.

[52] REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, v. 28, p. 91–99, 2015.

[53] GOODFELLOW, I. et al. Generative adversarial nets. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 2672–2680.

[54] DOSOVITSKIY, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[55] HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

[56] YU, Y. et al. A review of recurrent neural networks: Lstm cells and network architectures. *Neural Computation*, v. 31, n. 7, p. 1235–1270, 07 2019. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco_a_01199>.

[57] HOUDT, G. V.; MOSQUERA, C.; NáPOLES, G. A review on the long short-term memory model. *Artificial Intelligence Review*, v. 53, 12 2020.

[58] GERS, F.; SCHMIDHUBER, J.; CUMMINS, F. Learning to forget: Continual prediction with lstm. *Neural Computation*, v. 12, p. 2451–2471, 10 2000.

[59] GERS, F.; SCHRAUDOLPH, N.; SCHMIDHUBER, J. Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, v. 3, p. 115–143, 01 2002.

[60] WEN, X.; LI, W. Time series prediction based on lstm-attention-lstm model. *IEEE Access*, v. 11, p. 48322–48331, 2023.

[61] GERS, F. A.; SCHMIDHUBER, J.; CUMMINS, F. Learning to forget: Continual prediction with lstm. *Neural computation*, MIT Press, v. 12, n. 10, p. 2451–2471, 2000.

[62] FEDERATED Learning: Collaborative Machine Learning without Centralized Training Data. Https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. Accessed: 2022-13-08.

[63] MCMAHAN, B. et al. Communication-efficient learning of deep networks from decentralized data. In: PMLR. *Artificial intelligence and statistics*. [S.l.], 2017. p. 1273–1282.

[64] LI, T. et al. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, v. 2, p. 429–450, 2020.

[65] FANG, M. et al. *Local Model Poisoning Attacks to Byzantine-Robust Federated Learning*. 2021. Disponível em: <https://arxiv.org/abs/1911.11815>.

[66] MCMAHAN, B. et al. Communication-efficient learning of deep networks from decentralized data. In: PMLR. *Artificial Intelligence and Statistics*. [S.l.], 2017. p. 1273–1282.

[67] CHEN, M. et al. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intelligent Systems*, IEEE, v. 35, n. 4, p. 83–93, 2019.

[68] XU, J. et al. Federated learning for healthcare informatics. In: SPRINGER. *International Conference on Medical Image Computing and Computer-Assisted Intervention*. [S.l.], 2019. p. 119–127.

[69] GEYER, R. C.; KLEIN, T.; NABI, M. Differentially private federated learning: A client level perspective. In: *NIPS Workshop on Private Multi-Party Machine Learning*. [S.l.: s.n.], 2017.

[70] BONAWITZ, K. et al. Towards federated learning at scale: System design. In: *Proceedings of the 2nd SysML Conference*. [S.l.: s.n.], 2019.

[71] BAGDASARYAN, E. et al. How to backdoor federated learning. In: CHIAPPA, S.; CALANDRA, R. (Ed.). *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. PMLR, 2020. (Proceedings of Machine Learning Research, v. 108), p. 2938–2948. Disponível em: <https://proceedings.mlr.press/v108/bagdasaryan20a.html>.

[72] LIU, Y. et al. Trojaning attack on neural networks. In: *Network and Distributed System Security Symposium*. [s.n.], 2018. Disponível em: <https://api.semanticscholar.org/CorpusID:31806516>.

[73] MOHASSEL, P.; ZHANG, Y. Secureml: A system for scalable privacy-preserving machine learning. In: *2017 IEEE Symposium on Security and Privacy, SP 2017 - Proceedings*. United States: Institute of Electrical and Electronics Engineers Inc., 2017. (Proceedings - IEEE Symposium on Security and Privacy), p. 19–38. Funding Information: We thank Jing Huang from Visa Research for helpful discussions on machine learning, and Xiao Wang from University of Maryland for his help on the EMP toolkit. The work was partially supported by NSF grants 5245250 and

5246010. Publisher Copyright: © 2017 IEEE.; 2017 IEEE Symposium on Security and Privacy, SP 2017 ; Conference date: 22-05-2017 Through 24-05-2017.

[74] SHAFAHI, A. et al. Poison frogs! targeted clean-label poisoning attacks on neural networks. *CoRR*, abs/1804.00792, 2018. Disponível em: <http://arxiv.org/abs/1804.00792>.

[75] CHEN, X. et al. Targeted backdoor attacks on deep learning systems using data poisoning. *CoRR*, abs/1712.05526, 2017. Disponível em: <http://arxiv.org/abs/1712.05526>.

[76] BHAGOJI, A. N. et al. Analyzing federated learning through an adversarial lens. *CoRR*, abs/1811.12470, 2018. Disponível em: <http://arxiv.org/abs/1811.12470>.

[77] BIGGIO, B.; NELSON, B.; LASKOV, P. *Poisoning Attacks against Support Vector Machines*. 2013. Disponível em: <https://arxiv.org/abs/1206.6389>.

[78] JAGIELSKI, M. et al. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In: *2018 IEEE Symposium on Security and Privacy (SP)*. [S.l.: s.n.], 2018. p. 19–35.

[79] XIA, G. et al. Poisoning attacks in federated learning: A survey. *IEEE Access*, v. 11, p. 10708–10722, 2023.

[80] BLANCHARD, P. et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In: GUYON, I. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. v. 30. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf>.

[81] YIN, D. et al. Byzantine-robust distributed learning: Towards optimal statistical rates. In: DY, J.; KRAUSE, A. (Ed.). *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018. (Proceedings of Machine Learning Research, v. 80), p. 5650–5659. Disponível em: <https://proceedings.mlr.press/v80/yin18a.html>.

[82] MARCOZZI, M.; MOSTARDA, L. Analytical model for performability evaluation of practical byzantine fault-tolerant systems. *Expert Systems with Applications*, v. 238, p. 121838, 2024. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417423023400>.

[83] DENG, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, IEEE, v. 29, n. 6, p. 141–142, 2012.

[84] REYES-ORTIZ, J. et al. *Human Activity Recognition Using Smartphones*. 2012. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C54S4K.

[85] VITO, S. *Air Quality*. 2008. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C59K5F.

[86] JEBREEL, N. M. et al. Enhanced security and privacy via fragmented federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, Institute of Electrical and Electronics Engineers (IEEE), v. 35,

n. 5, p. 6703–6717, maio 2024. ISSN 2162-2388. Disponível em: <http://dx.doi.org/10.1109/TNNLS.2022.3212627>.

[87] BARUCH, G.; BARUCH, M.; GOLDBERG, Y. Little is enough: Circumventing defenses for distributed learning. In: *Advances in Neural Information Processing Systems*. [s.n.], 2019. v. 32. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2019/file/2cde6d232efa155027e7444cfc635309-Paper.pdf>.

[88] TAKAHASHI, K. et al. Confidence interval for micro-averaged f 1 and macro-averaged f 1 scores. *Applied Intelligence*, Springer, v. 52, n. 5, p. 4961–4972, 2022.

PAPERS PUBLISHED BY THE AUTHOR

Main Submission.

1. Blenda Oliveira Mazetto, Bruno Bogaz Zarpelão, **A Triad of Defenses to Mitigate Poisoning Attacks in Federated Learning**, Anais do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais, September/2024, SBC, 1-15, 000-0-000-00000-0, 10.5753/sbseg.2024.241712. (Qualis CC, A4), Awarded an Honorable Mention Certificate for a Full Paper.