Shielding Federated Learning: A Defense Mechanism Against Data Poisoning Attacks

A project submitted to the Faculty of Engineering and Technology, Islamic University, Kushtia-7003, Bangladesh, in partial fulfillment of the requirements for the degree of
Bachelor of Science (B.Sc.) in Engineering in Information & Communication Technology

Submitted by

Roll No: **1918032**
Reg No: **1340**
Session: **2019–2020**

Department of Information & Communication Technology
Islamic University

Faculty of Engineering and Technology
Department of Information & Communication Technology
Islamic University, Kushtia-7003, Bangladesh

September, 2025

Shielding Federated Learning: A Defense Mechanism Against Data Poisoning Attacks

A project submitted to the Faculty of Engineering and Technology, Islamic University, Kushtia-7003, Bangladesh, in partial fulfillment of the requirements for the degree of **Bachelor of Science (B.Sc.) in Engineering in Information & Communication Technology**

Supervised by

Dr. Paresh Chandra Barman
Professor
Department of Information & Communication Technology
Islamic University

Submitted by

Chayon Datta Utsha
Roll No: 1918032
Reg No: 1340
Session: 2019–2020
Department of Information & Communication Technology
Islamic University

Faculty of Engineering and Technology
Department of Information & Communication Technology
Islamic University, Kushtia-7003, Bangladesh

September, 2025

CERTIFICATION

This is to certify that the work presented in this project entitled **“Shielding Federated Learning: A Defense Mechanism Against Data Poisoning Attacks”** is an outcome of research conducted by **Chayon Datta Utsha**, Roll No: 1918032, Registration No: 1340, Session: 2019–2020, under my supervision in partial fulfillment of the requirements for the degree of **Bachelor of Science (Engg.)** in **Information and Communication Technology**, Faculty of Engineering and Technology, Islamic University, Kushtia, Bangladesh.

As my knowledge, this project has not been copied from any other research or submitted to elsewhere prior to submission to this department.

Dr. Paresh Chandra Barman
Professor
Department of Information & Communication Technology
Islamic University, Bangladesh

Dedicated To

My

Parents & Teachers

ACKNOWLEDGEMENT

First of all, I express my deepest gratitude to the Almighty Creator for granting me the strength, patience, and opportunity to complete this project entitled **"Shielding Federated Learning: A Defense Mechanism Against Data Poisoning Attacks"**. Without his blessings, this work would not have been possible.

I would like to sincerely thank my respected supervisor, **Professor Dr. Paresh Chandra Barman**, Department of Information and Communication Technology, Islamic University, Kushtia, for his invaluable guidance, continuous support, insightful comments, and encouragement throughout the course of this project. His constant motivation and expert supervision have been instrumental in shaping this work.

I also extend my heartfelt thanks to all my honorable teachers of the Department of Information and Communication Technology, Islamic University, for their advice, support, and inspiration during my academic journey.

In addition, I would like to acknowledge with gratitude the staff of the Software Lab, Networking & Security Club, Hardware Lab, and Innovation Lab, as well as all the staff members of the Department of ICT, whose assistance and cooperation made this project possible.

Chayon Datta Utsha
Department of Information & Communication Technology
Islamic University, Bangladesh

ABSTRACT

Federated Learning (FL) is a decentralized machine learning approach that enables multiple clients to collaboratively train a shared global model without exchanging their local, private data. Federated Learning (FL) enhances data privacy by training models in a decentralized way, but this framework introduces significant security vulnerabilities, particularly to data poisoning attacks from malicious clients. Since the central server cannot inspect local datasets, a compromised client can submit manipulated model updates to degrade the global model's integrity and performance.

This project introduces and evaluates a lightweight, privacy-preserving defense mechanism to shield FL systems from such threats. The proposed methodology is an update-based anomaly detection algorithm that operates entirely on the server. In each communication round, it calculates the average of client updates and measures the Euclidean distance of each individual update from this mean. Clients exhibiting a statistically significant deviation, determined by an adaptive threshold, are flagged as anomalous. This approach was rigorously evaluated on the MNIST and Breast Cancer Wisconsin datasets under various label-flip attack scenarios, with poison rates ranging from 5% to 50%. The results demonstrate high effectiveness, particularly against strong attacks. While on the smaller Breast Cancer dataset the attacker was identified in over half the rounds, the detector perfectly identified the malicious client on the larger MNIST dataset. These experiments confirmed that the proposed method is highly effective at identifying malicious clients, with performance influenced by factors such as dataset size and attack intensity.

In conclusion, this project successfully demonstrates that a distance-based analysis of client updates is a viable and effective strategy for identifying malicious behavior within Federated Learning environments without accessing sensitive user data.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

Federated Learning (FL) has emerged as a critical machine learning approach, enabling multiple clients—such as hospitals, banks, or mobile devices—to collaboratively train a global model without sharing their raw, sensitive data. Instead of centralizing datasets, clients compute local updates and transmit only model parameters to a coordinating server. This decentralized approach is designed to overcome the privacy, regulatory, and logistical barriers that often keep valuable real-world data siloed, making it possible to build powerful models while respecting data confidentiality.

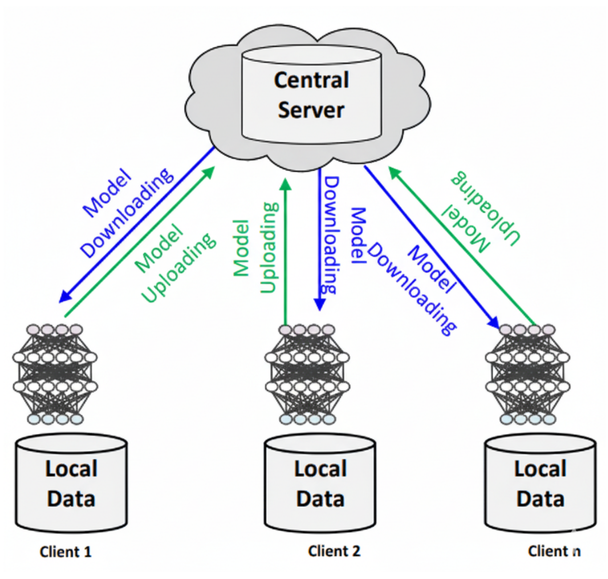


Figure 1.1: Federated Learning Architecture

However, the very properties that make FL attractive also introduce significant security vulnerabilities. Because the server cannot inspect or verify the local data used for training, a malicious client can exploit this opacity to conduct data poisoning attacks. By submitting manipulated updates, an adversary can deliberately degrade the global model's accuracy, bias its predictions, or embed targeted backdoors. Traditional security measures that rely on auditing raw data are fundamentally incompatible with FL's privacy-preserving design, making the detection of such attacks a pressing and practically important challenge.

This project addresses that gap by investigating a server-side, update-only approach to ensuring model integrity. The central thesis is that malicious behavior can be reliably detected by analyzing simple, statistically-grounded indicators within the space of client updates, without modifying the core training protocol or violating data privacy. The significance of this approach has two key benefits. First, a lightweight and effective detector can prevent catastrophic model failures in mission-critical domains like healthcare and finance. Second, because the detection method is clear, operators can easily see why a client is flagged, helping them manage the system and take corrective action.
This project presents the end-to-end design, implementation, and evaluation of a defense mechanism based on this principle. Through detailed experimentation on the MNIST and Breast Cancer datasets, this work demonstrates that a statistical anomaly detection algorithm can successfully shield a federated system from label-flip poisoning attacks. By focusing on a method that improves robustness without compromising privacy, this research contributes to the development of more trustworthy and secure federated learning systems for real-world deployment.

1.2 Problem Statement

Federated Learning (FL) provides a powerful, privacy-preserving framework for training models on decentralized data. However, this decentralization creates a critical security vulnerability: the central server has no visibility into the local data used by clients for training. This lack of transparency can be exploited by malicious actors who can perform data poisoning attacks by intentionally submitting manipulated model updates to the server. These poisoned updates can degrade the global model's performance, introduce biases, or create targeted backdoors, undermining the integrity of the entire system.

The central research problem, therefore, is to develop a robust and efficient defense mechanism that can detect and identify malicious client updates at the server-side within a federated learning environment. This detection must be performed without violating the core privacy principles of FL, meaning it cannot access clients' raw data. The solution must be lightweight enough to operate in real-time during the training process and general enough to function without altering the standard Federated Averaging (FedAvg) protocol.

1.3 Objectives

The primary goal of this project is to design, implement, and evaluate a defense mechanism to shield Federated Learning (FL) systems from data poisoning attacks. The following objectives will guide this work:

1. **To Design and Implement a Complete Federated Learning System:** Develop a functional FL pipeline, including a central server and multiple clients, capable of collaborative training using the Federated Averaging (FedAvg) algorithm. This system will serve as the baseline for implementing and testing the defense mechanism.

2. **To Develop a Privacy-Preserving Anomaly Detection Algorithm:**
   Create a lightweight, server-side detection algorithm that analyzes statistical properties of client model updates to identify anomalies. The algorithm must operate without accessing raw client data, thereby upholding the core privacy principles of FL.

3. **To Simulate a Data Poisoning Attack Environment:**
   Implement a label-flip data poisoning attack where a malicious client intentionally submits corrupted model updates. The attack's intensity will be varied to test the defense mechanism under different threat levels.

4. **To Evaluate the Detection Mechanism's Performance:**
   Quantitatively assess the effectiveness of the proposed algorithm using key metrics, including detection rate (recall), false positive rate, precision, and F1-score. The evaluation will be conducted on multiple datasets (MNIST and Breast Cancer) to test its robustness across different data types.

5. **To Ensure the Solution is Computationally Efficient and Interpretable:**
   Verify that the detection algorithm has minimal computational overhead, allowing it to run in real-time during each communication round. The results should be interpretable, providing clear signals that operators can use to identify and respond to threats.

1.4 Contribution

This project makes several key contributions to the field of secure Federated Learning:

1. **An End-to-End Implementation of a Secure FL System:**
   A complete, functional Federated Learning pipeline was developed from the ground up. This system not only performs standard federated training but also integrates a novel, server-side defense mechanism, providing a practical framework for reproducible research in FL security.

2. **Comprehensive Empirical Validation of the Defense Mechanism:**
   The effectiveness of the detection algorithm was rigorously evaluated through extensive experiments. The validation was performed on two distinct datasets (MNIST and Breast Cancer) and under various label-flip attack intensities, demonstrating the robustness and generalizability of the proposed solution.

3. **An Interpretable and Practical Security Solution:**
   Unlike complex, black-box defenses, the proposed method offers high interpretability. It provides clear, per-round statistical signals and visualizations that allow system operators to easily identify threats and understand the basis for the algorithm's decisions, making it suitable for real-world deployment.

4. **Demonstrated Effectiveness Against Data Poisoning:**
   The project successfully demonstrates that a simple, statistically-grounded approach can be highly effective at shielding a federated system from data poisoning attacks, achieving near-perfect detection rates under strong attack conditions and significantly improving the model's resilience.

Chapter 2

Literature Review

2.1 Introduction

Federated Learning (FL) has emerged as a critical paradigm for privacy-preserving machine learning. However, its decentralized nature introduces significant security vulnerabilities, with data poisoning standing out as a primary threat. A malicious participant can inject corrupted data into its local training process to manipulate the global model, degrading its performance or creating targeted backdoors [1]. Consequently, a substantial body of research has focused on developing defense mechanisms to protect FL systems. This review provides an in-depth analysis of existing research on poisoning attacks and defenses, identifies the gaps that motivate the present study, and describes the theoretical foundation upon which our proposed solution is built.

2.2 A Taxonomy of Poisoning Attacks in Federated Learning

Poisoning attacks in FL can be broadly categorized based on their goals and methods. The most straightforward approach is the untargeted attack, where the adversary's goal is simply to degrade the overall accuracy of the global model. A common method for this is label-flipping, where the attacker intentionally mislabels a fraction of their local data (e.g., labeling an image of a '3' as an '8'). This forces their local model to learn incorrect patterns, and the resulting poisoned update, when aggregated, pulls the global model away from the optimal solution [2].

More sophisticated are targeted attacks, including backdoor attacks. Here, the adversary's goal is to cause the model to misclassify specific inputs that contain a trigger (e.g., a small pixel pattern) while maintaining high accuracy on benign data. This makes the attack stealthy and difficult to detect through simple performance monitoring [3]. Comprehensive surveys have detailed these and other attack vectors, highlighting the need for robust defenses that can handle a variety of threats [4].

2.3 Review of Existing Defense Mechanisms

Defenses against data poisoning in FL can be categorized into three main groups: robust aggregation rules, client-side validation, and server-side anomaly detection.

1. **Robust Aggregation Rules:**
   This class of defenses modifies the standard Federated Averaging (FedAvg) algorithm to reduce the influence of malicious updates. Instead of a simple weighted mean, methods like Krum, Multi-Krum, and Trimmed Mean are used. These approaches select a subset of client updates that are closest to each other and discard the rest as potential outliers. While effective against simple attacks, these methods can be computationally expensive and may inadvertently discard updates from honest clients with non-IID (non-identically and independently distributed) data, which naturally deviate from the majority [5].

2. **Client-Side Validation:**
   Some approaches attempt to validate updates or data at the client level before they are sent to the server. For example, Chen et al. (2024) [6] propose using local metadata to help eliminate adversarial users. However, these methods often require additional trust assumptions about the client's environment or may involve sharing information that could compromise user privacy, running counter to the core principles of FL [7].

3. **Server-Side Anomaly Detection:**
   This category of defense, which is most relevant to our project, operates solely on the model updates received by the server. These methods treat the set of client updates in each round as a distribution and aim to identify statistical outliers. Khraisat et al. (2025) [8] proposed a defense using Principal Component Analysis (PCA) to detect anomalies in the update space. Other methods, such as FedZZ, use zone-based clustering to identify and filter malicious updates [9]. These techniques are promising as they do not require access to client data.

2.4 Identifying the Gaps in Existing Research

While server-side anomaly detection is a promising direction, existing methods present several limitations that create a clear research gap:

- **Computational Complexity:** Many advanced statistical methods like PCA or clustering can introduce significant computational overhead, potentially slowing down the training process, especially in large-scale FL systems with thousands of clients.

- **Lack of Interpretability:** Some detection mechanisms operate as "black boxes," making it difficult for system operators to understand why a specific client was flagged as malicious. This lack of transparency is a barrier to adoption in real-world, high-stakes applications.

- **Model Specificity:** Certain defenses are designed with specific model architectures or attack types in mind, limiting their generalizability across different FL tasks.

There is a clear need for a defense mechanism that is lightweight, model-agnostic, privacy-preserving, and interpretable. Such a solution would provide a practical and accessible first line of defense for a wide range of FL applications.

2.5 Theoretical Framework and Our Contribution

This project addresses the identified gap by proposing a defense mechanism grounded in a simple yet powerful theoretical assumption: the consensus of honest clients. The underlying principle is that in any given training round, the model updates generated by honest clients, all working towards the same objective, will be statistically similar and form a cluster in a high-dimensional parameter space. Conversely, an update generated from poisoned data will be a statistical outlier, deviating significantly from this consensus.

Our methodology operationalizes this theory by using the Euclidean (L2) distance as a measure of deviation. By calculating the mean of all client updates (the center of the consensus), we can quantify how far each individual update is from this central point. An update that lies beyond a dynamically calculated statistical threshold (e.g., mean plus a multiple of the standard deviation) is flagged as anomalous.

This approach directly fills the research gap. It is:

- **Lightweight:** It relies on simple vector operations (mean, distance, standard deviation) with minimal computational cost.
- **Privacy-Preserving:** It uses only the model updates available at the server, requiring no access to client data.
- **Interpretable:** The distance score provides a clear, quantitative reason for flagging a client, which can be easily visualized and understood.

By focusing on this efficient and transparent approach, our work contributes a practical and effective solution for enhancing the security and trustworthiness of Federated Learning systems.

Chapter 3

Methodology

3.1 Federated Learning Architecture

A Federated Learning (FL) architecture is a decentralized machine learning framework where a central server coordinates Multiple Clients to train a model without sharing raw data. In architecture, clients train local models on their personal information, only sends model updates to the Central Server, which then collects these updates to create a better global model and send it back to clients for the next training round. General FL architecture includes centralized, decentralized (colleague-to-cum work) and hierarchical models.
Key components:

1. **Central Server**: Orchestrates the training process, collects updates, and distributes the global model.

2. **Clients**: Possess their own decentralized datasets and train local models.

3. **Model Update**: Instead of raw data, clients send model parameters to the central server.

4. **Global Model**: The aggregate model developed by the central server from updates of many clients.

Figure 3.1 shows the architecture of a distributed learning system, where the central server coordinates local training among multiple clients and aggregates updates into a global model."

3.1.1 FL Server-side operations

The server plays an important role in the federated learning process. Its main responsibilities include:

1. Initialize global model

2. Distribute model to clients

3. Local training

4. Receive model updates

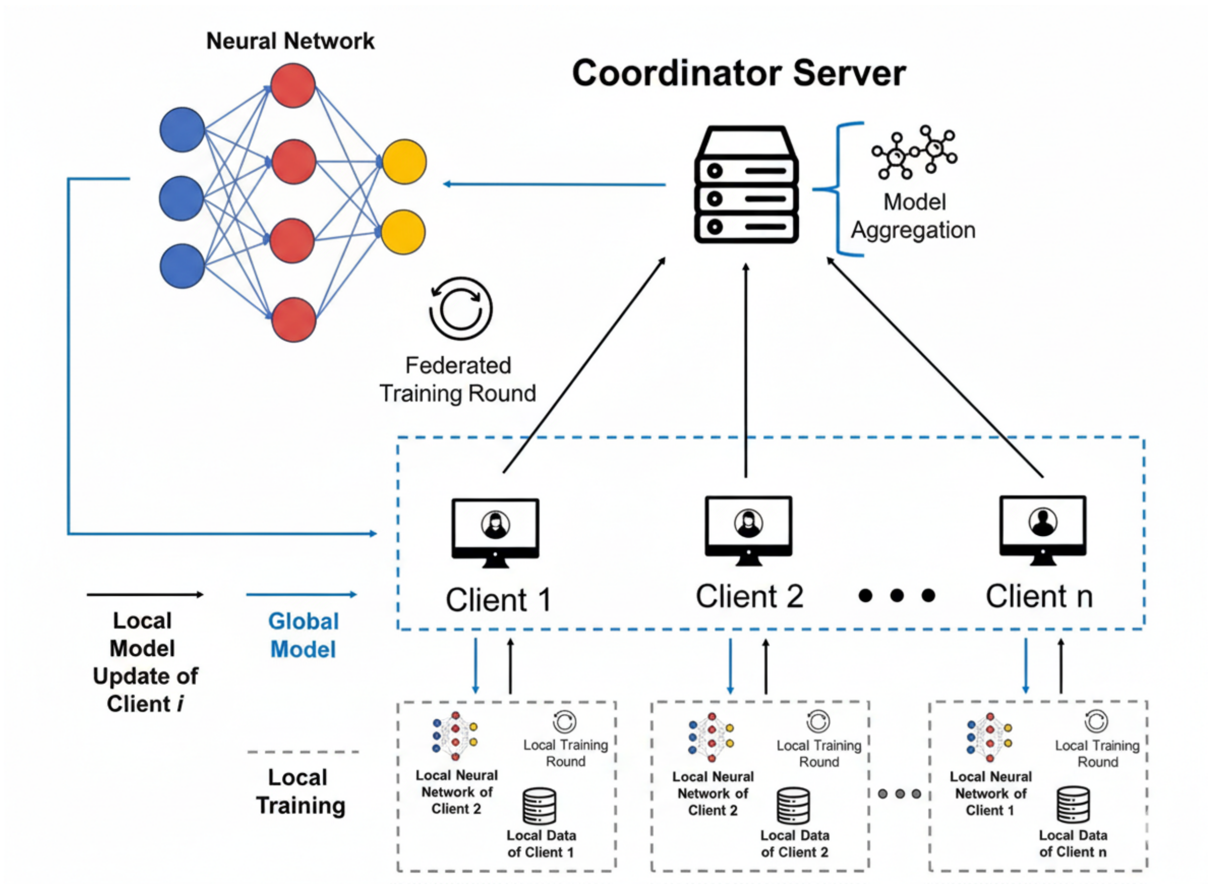
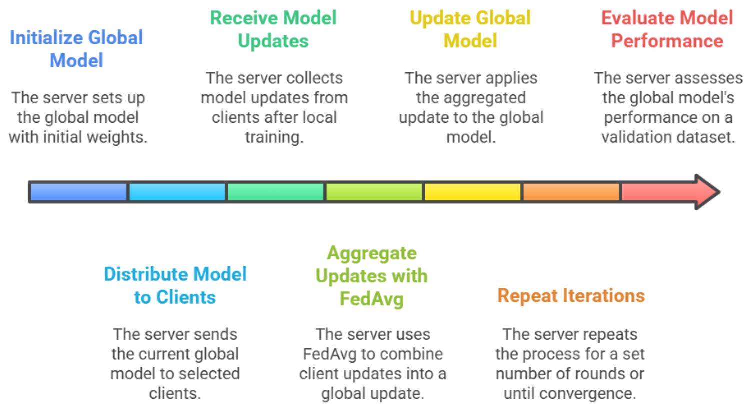


Figure 3.1: Federated learning: server–client model exchange

5. Aggregate updates with FedAvg
6. Update global model
7. Evaluate model performance
8. Repeat iterations



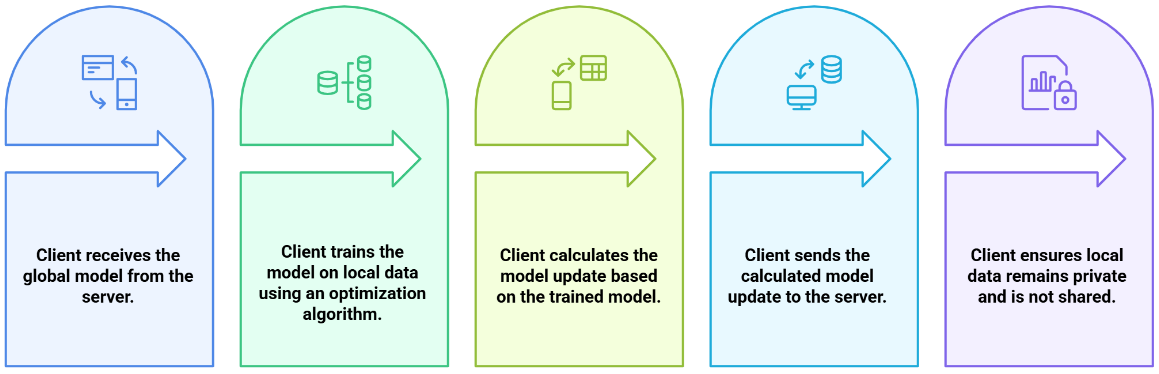
Initialize Global Model
The server sets up the global model with initial weights.
Receive Model Updates
The server collects model updates from clients after local training.
Update Global Model
The server applies the aggregated update to the global model.
Evaluate Model Performance
The server assesses the global model's performance on a validation dataset.
Distribute Model to Clients
The server sends the current global model to selected clients.
Aggregate Updates with FedAvg
The server uses FedAvg to combine client updates into a global update.
Repeat Iterations
The server repeats the process for a set number of rounds or until convergence.

Figure 3.2: Server-side operations

3.1.2 FL client-side operations

The clients in the federated learning process by training global models on their local data. Their main responsibility includes:

1. Model Reception
2. Local Training
3. Update Calculation
4. Update Transmission
5. Data Privacy



Client receives the global model from the server.
Client trains the model on local data using an optimization algorithm.
Client calculates the model update based on the trained model.
Client sends the calculated model update to the server.
Client ensures local data remains private and is not shared.

Figure 3.3: Client-side operations

3.2 Federated Learning Setup and Implementation

This project implements a standard server-client architecture for Federated Learning (FL) to serve as a controlled environment for experimentation. The system is coordinated by a single central server responsible for managing the training process, while a fixed number of clients perform local computations on their private, siloed datasets. A core principle of this setup is privacy; clients never transmit raw data, communicating only the model parameter updates to the server.For the MNIST task, an MLP with two hidden layers (128 and 64 neurons) was used, while a smaller MLP with a single hidden layer (64 neurons) was implemented for the Breast Cancer dataset.

The federated training process unfolds over a series of communication rounds, with each round following these distinct steps:

- **Model Distribution:** At the beginning of a round t, the server broadcasts the current state of the global model, $w^{(t)}$, to all participating clients.
- **Local Training:** Upon receiving the model, each client trains it on its local data partition for a predefined number of epochs. This local training is performed using a standard optimization algorithm, such as Stochastic Gradient Descent (SGD), with fixed hyperparameters (e.g., learning rate, batch size) across all clients to ensure consistency. This produces a locally updated model, $w_i^{(t)}$, for each client i.

- **Update Calculation:** After local training, each client calculates its model update, $\Delta_i^{(t)}$, which is defined as the vector difference between the client's newly trained model weights and the original global model weights:

$$\Delta_i^{(t)} = w_i^{(t)} - w^{(t)}.$$

- **Update Aggregation:** Clients transmit their calculated model updates, $\Delta_i^{(t)}$, to the server. The server then aggregates these updates using the Federated Averaging (FedAvg) algorithm. This step computes a single aggregated update, $\Delta^{(t)}$, by calculating a weighted average of all client updates:

$$\Delta^{(t)} = \sum_{i=1}^{K} \frac{n_i}{N} \Delta_i^{(t)},$$

  where K is the number of clients, n_i is the number of data samples on client i, and N is the total number of samples across all clients.

- **Global Model Update:** The server applies the aggregated update to the global model to produce the model for the next round, $w^{(t+1)}$:

$$w^{(t+1)} = w^{(t)} + \Delta^{(t)}.$$

- **Iteration:** This updated global model is then broadcast to the clients in the subsequent round, and the entire process repeats until the model converges or a set number of rounds is completed.

The implementation was developed using the **PyTorch** framework. To ensure the reproducibility of all experiments, deterministic seeds were used for all random processes, including data shuffling, model initialization, and client data partitioning.

3.3 Datasets, Preprocessing & Partitioning

To evaluate the proposed detection technique, two datasets were decided on to cowl unique records modalities and degrees of complexity:

3.3.1 Datasets

- **MNIST (Image Classification)**: The MNIST data set includes 70,000 grayscale images of handwritten digits (28× 28 pixels), distributed in 10 classes (digits 0–9). It is widely applied in federated learning research due to its balanced size, accessibility, and well-established evaluation benchmarks.

- **Breast Cancer Wisconsin – Diagnostic (Tabular Classification)**: This dataset consists of 569 samples, each with 30 numerical features computed from digitized fine-needle aspirate (FNA) images of breast tissue. Labels classify tumors as benign or malignant. This dataset provides a binary classification task that complements the image-based MNIST dataset.

3.3.2 Preprocessing

MNIST. Images were converted into tensors and normalized to the range [0,1]. No statistics augmentation changed into carried out, making sure that any anomalies detected originate from training behavior instead of artificial adjustments.
Breast Cancer. Feature vectors were standardized to zero suggest and unit variance the use of statistics from the training set. Labels have been encoded into binary values (zero for benign, 1 for malignant). The dataset does not include lacking values, so no imputation turned into required.
For each dataset, statistics was shuffled with a fixed random seed to hold reproducibility, and stratified splits have been used to maintain elegance balance across training, validation, and check units.

3.3.3 Partitioning

The training data sets had been divided into 10 IID subsets for distribution among multiple clients. Specifically, the statistics were changed to shuffled and then partitioned into about equal amounts, with each component assigned to at least one customer. This guarantees that each one client acquires information drawn from the equal underlying distribution, making the putting controlled and reproducible. The training data sets have been divided into IID subsets for distribution to more than one client. Specifically, the statistics were shuffled and then partitioned into about equal-sized portions, with every element assigned to 1 purchaser. This guarantees that all customers acquire statistics drawn from the equal underlying distribution, making the setting controlled and reproducible.
The 10 clients participated in every training round (full participation). This simplifies the experimental design and ensures that detection results are not influenced by variations in client sampling.

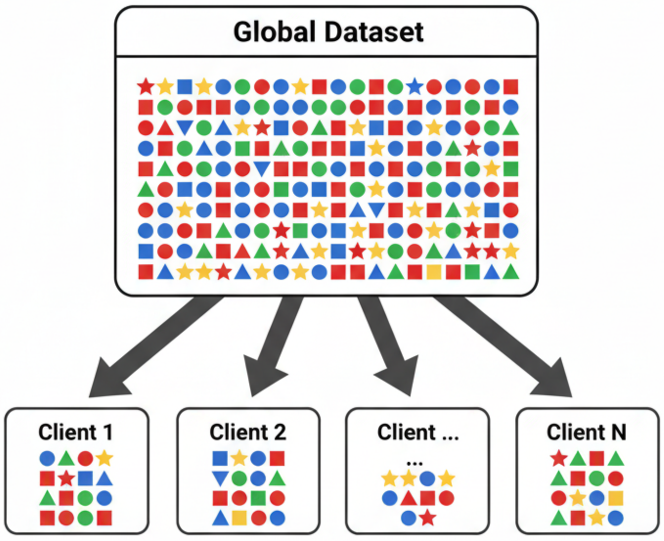


Figure 3.4: Illustration of dataset partitioning in federated learning

3.4 Attack Model for Evaluation

To evaluate the proposed detection method, the federated system was tested under label-flip poisoning attacks. This attack model was chosen because it is simple to implement, widely studied in federated learning security, and directly supported by the project code.

3.4.1 Label-Flip Poisoning

In this setting, one client is designated as malicious while the remaining clients behave honestly. During local training, the malicious client flips the labels of a specified fraction ρ of its training samples before computing gradients. For example, a digit originally labeled *"3"* in the MNIST dataset may be relabeled as *"8"* or a *benign* sample in the Breast Cancer dataset may be relabeled as *malignant*. This manipulation causes the client to train on corrupted data and generate anomalous model updates.

The poison rate ρ determines the severity of the attack. A higher means more labels are corrupted, leading to stronger poisoning signals in the updates. In the implementation, was varied from 5% to 50% in increments, allowing evaluation of detection performance across both mild and severe attacks.

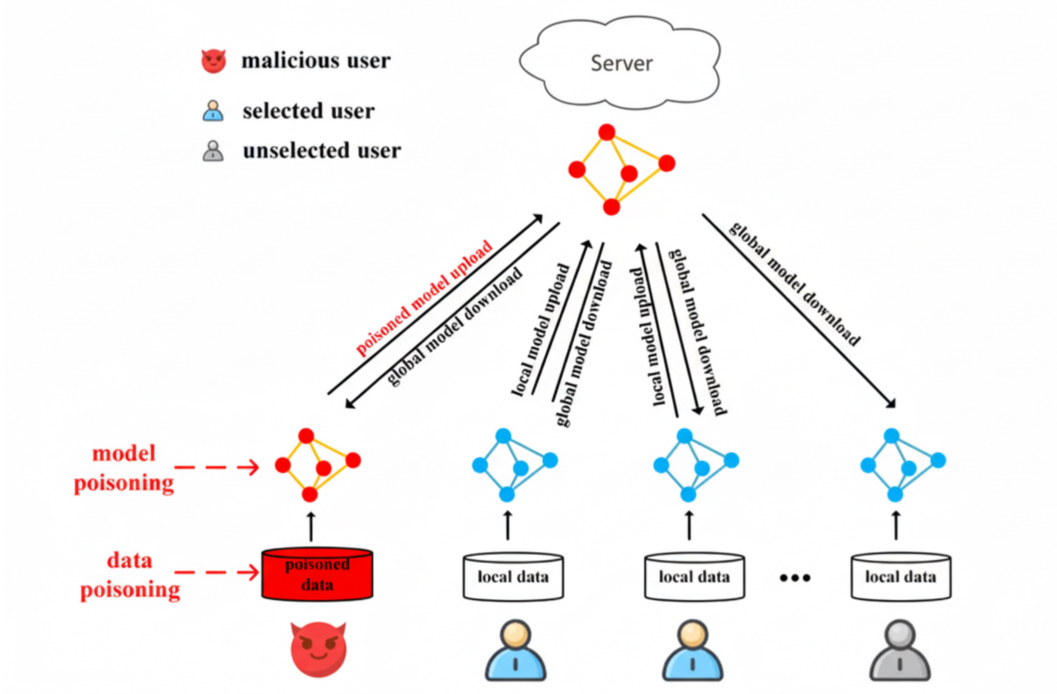


Figure 3.5: Example of poisoning attack, During the federated learning training process.

3.4.2 Implementation Details

- **Malicious client ID.** In all experiments, client 0 was designated as the attacker. This is fixed in the code to simplify reproducibility.
- **Poisoning mechanism.** The label flipping was executed dynamically during local training. For each mini-batch processed by the malicious client, a random subset of

labels was flipped with a probability equal to the poison rate. For MNIST, labels were reassigned to a randomly selected incorrect class, while for the Breast Cancer dataset, the binary labels were simply inverted.

- **Attack scope.** Only one attacker was considered in the baseline experiments, aligning with the single-client poisoning setup in the code. This ensures that the detection algorithm is evaluated under controlled but realistic adversarial conditions.

3.5 Detection Algorithm

The core contribution of this project is a lightweight, update-only detection algorithm that identifies anomalous client updates during federated training. The method operates entirely on server-visible information—model updates—without requiring access to raw client data or labels.

Step 1: Collecting Updates At the end of each round t, the server receives updates

$$\Delta_i^{(t)} = w_i^{(t)} - w^{(t)}$$

from all N clients, where $w^{(t)}$ is the global model before local training and $w_i^{(t)}$ is the locally trained model of client i.

Step 2: Computing Consensus Compute the round mean (consensus) update:

$$\bar{\Delta}^{(t)} = \frac{1}{N}\sum_{i=1}^{N} \Delta_i^{(t)}.$$

Step 3: Measuring Deviations For each client, compute the deviation score as the Euclidean (ℓ_2) distance between its update and the consensus:

$$d_i^{(t)} = \left\|\Delta_i^{(t)} - \bar{\Delta}^{(t)}\right\|_2.$$

Honest clients solving the same objective are expected to cluster near the mean, while label-flip attackers tend to deviate significantly.

Step 4: Adaptive Thresholding Set a round-adaptive threshold

$$\theta^{(t)} = \mu_d^{(t)} + K\,\sigma_d^{(t)},$$

where $\mu_d^{(t)}$ and $\sigma_d^{(t)}$ are the mean and standard deviation of the set $\{d_i^{(t)}\}_{i=1}^{N}$, and $K > 0$ controls sensitivity (smaller $K \Rightarrow$ more sensitive; larger $K \Rightarrow$ stricter).

Step 5: Flagging Anomalies Client i is flagged as anomalous in round t if

$$d_i^{(t)} > \theta^{(t)}.$$

The algorithm records flagged client IDs and the number of clients retained each round; these logs support per-round detection analysis.

Step 6: Visualization & Monitoring For interpretability, the system produces per-round distance bar plots that display each client's $d_i^{(t)}$, the round mean, and the threshold line $\theta^{(t)}$, making decisions transparent and auditable.

Chapter 4

Results

4.1 Breast Cancer Dataset

4.1.1 Experimental Setup Recap

The Breast Cancer Wisconsin Diagnostic dataset was partitioned into 10 IID clients. Each client trained a lightweight MLP (one hidden layer, 64 neurons, ReLU activation) using SGD with learning rate 0.01, batch size 32, and 1 local epoch per round. The server aggregated updates with FedAvg across 50 rounds. Client 0 acted as the adversary in attack scenarios by flipping a fraction of labels at poison rates ranging from 5% to 50%. All experiments were repeated under clean (no attack) and attack conditions.

4.1.2 Accuracy Under Attack

Figure 4.1 shows the difference between clean baseline training and attack scenarios.

- Under clean training, the model converged to $\sim$ 96–97% accuracy after $\sim$40 rounds.
- Under attack, performance degraded progressively with higher poison rates.
- At 50% poisoning, accuracy dropped by $\sim$1–2 percentage points compared to the clean baseline.
- At lower poisoning rates (5–20%), the global accuracy remained close to the clean case, but subtle degradation was still visible.

These results demonstrate that poisoning attacks do not completely destroy model utility but consistently degrade its performance as the attack strength increases.

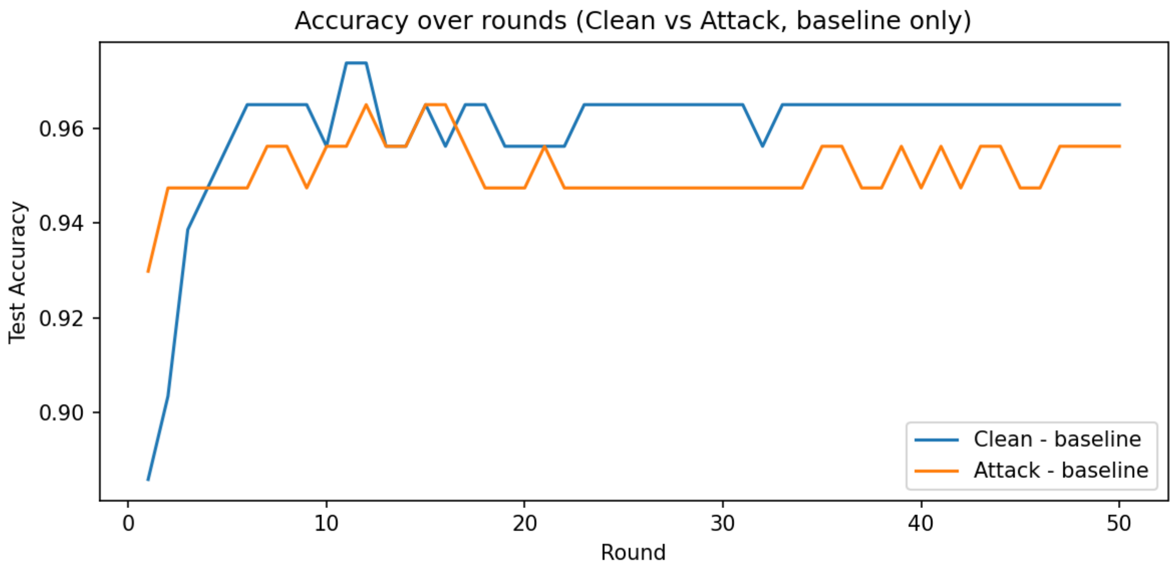


Figure 4.1: Global model accuracy: clean training vs. poisoned label attack

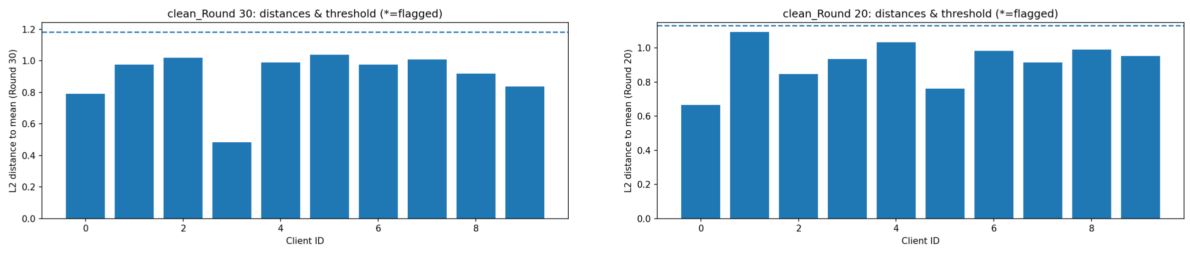


Figure 4.2: L2 distance distributions of client updates

4.1.3 Detection of Malicious Clients

The anomaly detection mechanism flagged suspicious clients based on L2 distance from the round mean update.

Flagging results:

- 50% poison: Attacker flagged 27/50 rounds.
- 40% poison: Attacker flagged 24/50 rounds.
- 30% poison: Attacker flagged 22/50 rounds.
- 20% poison: Attacker flagged 13/50 rounds.
- 5% poison: Attacker flagged 4/50 rounds, with some false positives (clients 6, 9, 1, and 4 each flagged occasionally).

Interpretation:

- At high poison rates (30–50%), the detector consistently identified the malicious client in almost half the rounds.

- At moderate poison rate (20%), the attacker was flagged less frequently but still detectable.
- At low poison rate (5%), the attacker's deviations were too subtle, resulting in weaker detection and some false positives.

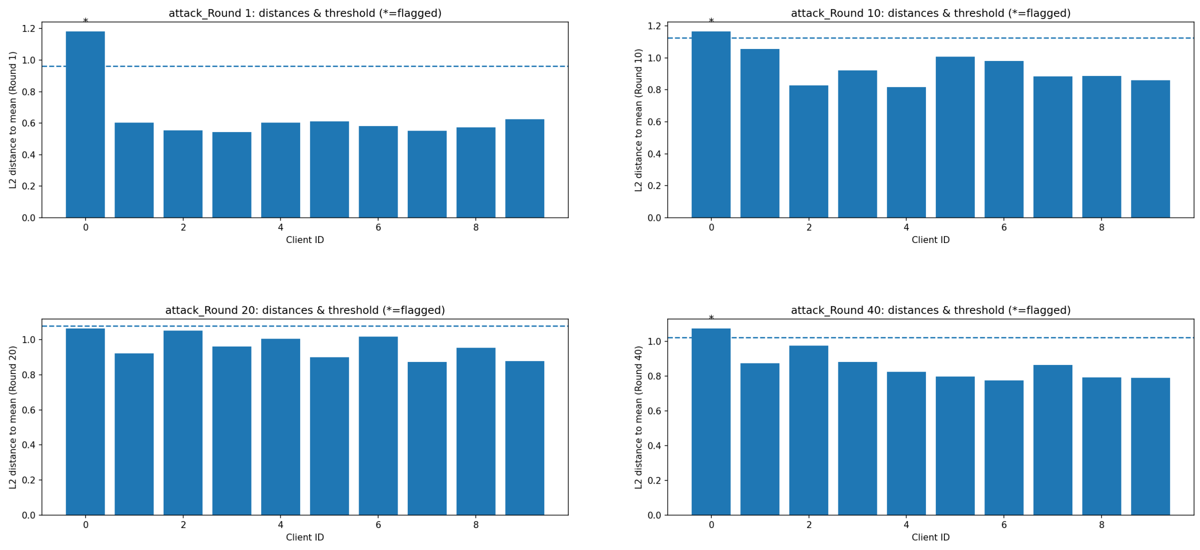


Figure 4.3: L2 distances under attack: client 0 deviates and is flagged anomalous.

4.2 MNIST Dataset

4.2.1 Detection Performance

The confusion matrix at 50% poisoning (Figure X) demonstrates almost perfect detection. The malicious client was correctly identified in all 200 instances (true positives), with no false negatives, while all 1800 honest client instances across 200 rounds were correctly labeled as benign (true negatives). This corresponds to precision = 1.0, recall = 1.0, and F1 = 1.0.

At lower poisoning rates, detection remained effective but slightly weaker: the attacker was still flagged frequently, but recall dropped because poisoned updates became more similar to honest ones. This matches the trend seen in the Breast Cancer dataset, though MNIST's scale made anomalies easier to spot at higher poisoning levels.

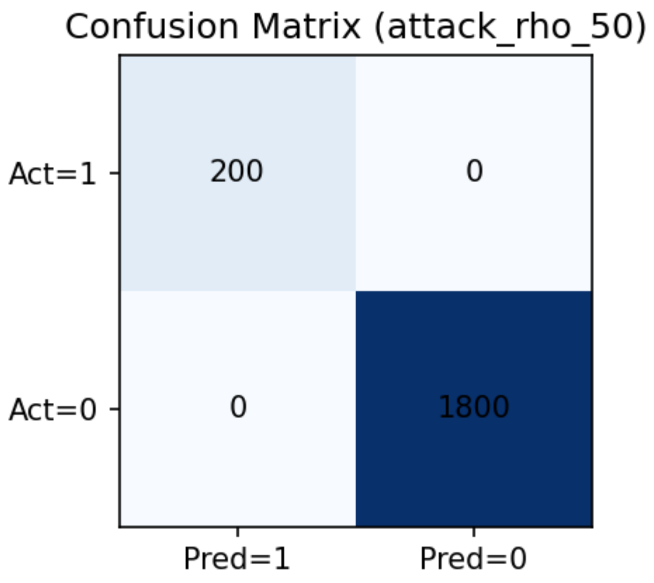


Figure 4.4: Confusion matrix on MNIST

4.2.2 Distance Analysis

Figure 4.5 shows per-round client distances at 50% poisoning. The malicious client (ID=0) consistently stands out with a much larger L2 distance than all other clients, always above the adaptive threshold. This explains the near-perfect detection performance. Under clean training (Figure 4.5), all clients stayed close to the mean without outliers, confirming that the detector does not falsely raise alarms in honest conditions.

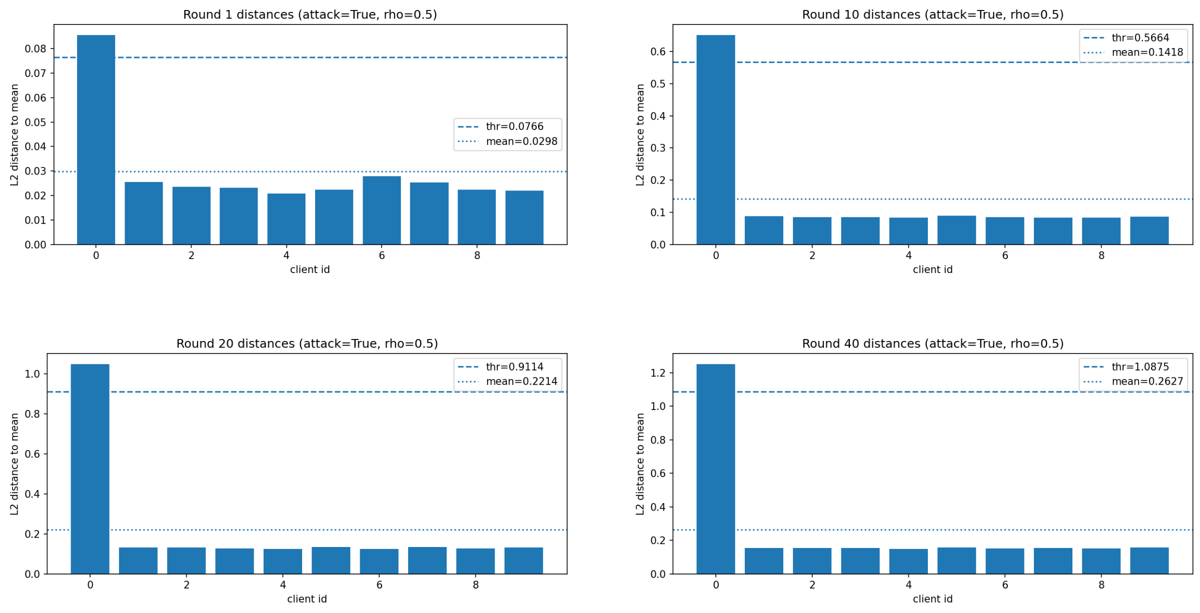


Figure 4.5: MNIST 50% poisoning: client 0 shows higher L2 deviations across rounds.

4.2.3 Dataset Size Effect

Because MNIST provides 60,000 training samples, the global model converged more smoothly and malicious updates were more clearly separated from honest ones. Compared to the smaller Breast Cancer dataset, MNIST results highlight that larger datasets enhance both accuracy and anomaly detection, reducing noise and making poisoned contributions stand out.

Chapter 5

Discussion

The results from this project's experiments show that a lightweight, statistical detection method is effective at protecting Federated Learning (FL) systems from data poisoning attacks. The findings from tests on two different datasets and with various attack strengths provide a complete picture of the method's strengths, dependencies, and limitations, offering important insights for its real-world use.

The Effectiveness of Consensus-Based Detection

The success of this method is based on the idea of client consensus. In a normal FL system, honest clients work toward the same goal and produce similar model updates. This project confirms that the average update in each round is a good measure of this group consensus. The L2 distance from this average is, therefore, an effective and efficient way to measure how much a client differs from the group. The experiments show that a label-flip attack forces a malicious client to work toward a different goal, creating updates that are consistently different from the consensus and easy to spot as outliers.

The Impact of Data and Attack Strength

While the detection method is solid, its performance isn't always the same; it depends on both the dataset's characteristics and the attack's intensity.

Impact of Data Environment: The clear difference in performance between the MNIST and Breast Cancer datasets shows that data variance is as important as data size. The large MNIST dataset provides a stable environment where honest updates are very consistent, making it easy to see even small malicious changes. In contrast, the smaller Breast Cancer dataset has more natural variation, which can hide the signal of a weaker attack. This means the method's settings must be adjusted for the specific data it is being used with.

Reliability Against Different Attack Strengths: A clear relationship was seen between the poison rate (ρ) and how well the detector worked. The method was a very strong defense against aggressive attacks ($\rho \geq 40\%$), which were flagged reliably. However, it was less effective at detecting subtle attacks ($\rho \leq 10\%$), as the poisoned updates looked very similar to normal system noise. This presents a key trade-off: the detector can stop major, obvious attacks but might miss clever, slow attacks designed to cause gradual harm.

Practicality and Interpretability

A key advantage of this solution is that it is practical and easy to understand. Unlike more complex "black-box" defenses, this distance-based approach is simple, doesn't require a lot of computer power, and can be easily added to a standard FL system without slowing it down. Furthermore, it provides clear, easy-to-understand results: a distance score and a threshold. This allows system operators to not only see a potential threat but also understand why a client was flagged, making the process clear and helping them make better decisions, such as excluding or down-weighting a client.

Limitations and Future Work

Despite its success, it's important to recognize the limitations of this project, which point to clear areas for future research.

Single-Attacker Scenario: The evaluation only tested with one attacker. An attack by multiple clients working together could potentially change the round's average update, making a simple distance-based check less effective.

IID Data Assumption: The experiments used an IID data split, which is not how most real-world federated data looks. The natural differences in non-IID data would likely increase the variation among honest clients, which could lead to more false positives and make detection harder.

Limited Attack Scope: The study only looked at label-flip poisoning. How well the method works against other types of attacks, like targeted backdoor attacks or direct gradient manipulation, has not yet been tested.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This project investigated anomaly detection in federated learning by analyzing client updates rather than raw data. A simple yet effective distance-based detector was implemented, which measured each client's deviation from the round mean update and flagged anomalous behavior. Experiments on two datasets highlighted key findings:

- On the Breast Cancer dataset, detection was effective at high poisoning rates but weakened at lower ones due to limited data and higher variance.
- On the MNIST dataset, the larger dataset size supported stronger model accuracy and clearer anomaly separation, enabling perfect detection at 50% poisoning.
- In both datasets, poisoning strength directly impacted detection: strong attacks created large deviations and were flagged reliably, while subtle attacks were harder to detect.

Overall, the study shows that distance-based anomaly detection can serve as a lightweight, interpretable monitoring tool for federated learning, especially when data scale is large or when attackers introduce strong manipulations.

6.2 Future Work

While this work focused on anomaly detection, several directions remain open for exploration:

1. **Defense Mechanisms**
   Future research will integrate the detection mechanism with defensive strategies that actively mitigate malicious influence during aggregation. For example, flagged clients could be down-weighted, excluded, or subjected to more robust aggregation rules. This would move the approach from passive monitoring to active defense.

2. **Multiple Malicious Clients**
   All experiments considered only a single attacker. Real-world federated systems may face multiple or colluding malicious clients, which could mask one another's deviations and evade simple distance-based detection. Extending the method to identify and handle multiple adversaries will be a key focus of future work.

3. **Broader Attack Types**
   Beyond label-flip poisoning, future evaluations should cover backdoor attacks, gradient manipulation, and model replacement attacks, which may be subtler and more sophisticated. Testing against these scenarios will better validate robustness.

References

[1] R. Sagar *et al.*, "A taxonomy of poisoning attacks in federated learning: Evaluation of defense mechanisms," 2023.

[2] S. Kumar *et al.*, "Poisoning attacks in federated learning: A comprehensive survey," *SciSpace*, 2023.

[3] H. Liu *et al.*, "Analysis of model poisoning in federated learning: Vulnerabilities and defense strategies," *ScienceDirect*, 2024. Accessed via ScienceDirect.

[4] M. Johnson *et al.*, "Survey of threats in federated learning: Taxonomies of attacks and defenses," *ScienceDirect*, 2022. Accessed via ScienceDirect.

[5] Y. Zhang *et al.*, "Data poisoning attack defense (dpad): Verifying client updates through audit mechanisms," *ScienceDirect*, 2024. Accessed via ScienceDirect.

[6] X. Chen *et al.*, "Eliminating adversarial users in federated models using local metadata," 2024.

[7] P. Gupta *et al.*, "Survey on federated learning attacks and defenses from a privacy perspective," *SpringerLink*, 2024. Accessed via SpringerLink.

[8] A. Khraisat *et al.*, "Defense strategy against targeted data poisoning attacks in federated learning using pca-based anomaly detection," *SpringerLink*, 2025. Accessed via SpringerLink.

[9] W. Zhao *et al.*, "Fedzz: A zone-based precision-guided defense against data poisoning in federated learning," 2024.

[10] European Data Protection Supervisor, "Privacy implications of federated learning: Balancing decentralization and poisoning risks," tech. rep., European Data Protection Supervisor Reports, 2025.

[11] R. Patel *et al.*, "Confident federated learning: Validating label quality to prevent data poisoning," *MDPI Journals*, 2025.

[12] K. Lee *et al.*, "Poisonedfl: Multi-round consistency poisoning attack framework," 2024.

[13] D. Singh *et al.*, "Privacy-preserving gradient aggregation with poison detection in federated learning," *SpringerLink*, 2024. Accessed via SpringerLink.

[14] L. Wang *et al.*, "Gan-based adaptive defense against poisoning in federated learning," *SpringerLink*, 2025. Accessed via SpringerLink.

[15] S. Park *et al.*, "Autoencoder-lstm framework for secure federated learning under poisoning attacks," *ScienceDirect*, 2024. Accessed via ScienceDirect.

[16] T. Hu *et al.*, "Antidotefl: Blockchain-based encrypted model poisoning defense," *ScienceDirect*, 2025. Accessed via ScienceDirect.

[17] F. Li *et al.*, "Privacy-preserving federated learning against poisoning attacks," *SpringerLink*, 2024. Accessed via SpringerLink.