Enhancing Federated Learning Security with a Defense Framework Against Adversarial Attacks in Privacy-Sensitive Healthcare Applications

Frederick Ayensu, Claude Turner, Isaac Osunmakinde
Department of Computer Science, Norfolk State University, Virginia, USA

***Abstract*—Federated learning (FL) is a cutting-edge method of collaborative machine learning that lets organizations or companies train models without exchanging personal information. Adversarial attacks such as data poisoning, model poisoning, backdoor attacks, and man-in-the-middle attacks could compromise its accuracy and reliability. Ensuring resistance against such risks is crucial as FL gets headway in fields like healthcare, where disease prediction and data privacy are essential. Federated systems lack strong defenses, even though centralized machine learning security has been extensively researched. To secure clients and servers, this research creates a framework for identifying and thwarting adversarial attacks in FL. Using PyTorch, the study evaluates the framework's effectiveness. The baseline FL system achieved an average accuracy of 90.07%, with precision, recall, and F1-scores around 0.9007 to 0.9008, and AUC values of 0.95 to 0.96 under benign conditions. With AUC values of 0.93 to 0.94, the defense-enhanced FL system showed remarkable resilience and maintained dependable classification (precision, recall, F1-scores ~0.8590–0.8598), despite a 4.1% accuracy decline to 85.97% owing to security overhead. With an 84.33% attack detection rate, 99.32% precision, 96.62% accuracy and a low false positive rate of 0.15%, the defense architecture performed exceptionally well in adversarial attacks. Trade-offs were identified via latency analysis: the defense-enhanced system stabilized at 54 to 56 seconds, while the baseline system averaged 13-second rounds. With practical implications for safe, robust machine learning partnerships, these findings demonstrate a balance between accuracy, efficiency and security, establishing the defense-enhanced FL system as a reliable option for privacy-sensitive healthcare applications.**

Keywords*—Federated learning; machine learning; privacy; adversarial attacks; defense framework; global model; healthcare; disease prediction

I. Introduction

Federated Learning (FL) is a collaborative machine learning technique that allows decentralized training while maintaining data security [1, 2]. FL is vulnerable to adversarial attacks that can compromise the integrity of the model, its performance, and the extraction of sensitive information [3]. Defense frameworks, equipped with robust aggregation methods, anomaly detection, and privacy-preserving mechanisms, fortify FL systems against these attacks [4]. By integrating these frameworks, comprehensive solutions can effectively address a wide range of threats simultaneously [5, 6]. Despite efforts, dynamic environments and evolving attacks make it difficult to develop a secure FL system.

A critical challenge in federated learning (FL) is achieving a balance between security, privacy, and model performance, particularly in privacy-sensitive healthcare, where data protection is paramount [7, 8]. Adversarial attacks, such as model poisoning, data tampering, backdoor attacks, and man-in-the-middle attacks, can compromise model integrity and performance, yet existing FL systems often lack robust defenses to counter these threats while maintaining scalability and quality [9, 10]. The scarcity of empirical research on secure FL in healthcare further complicates its adoption, as evolving cyber threats demand adaptable, scalable solutions for real-world deployment.

The primary objective of this research is to develop and evaluate a defense framework that ensures the reliability and safety of FL systems, particularly in the medical field. The research explores various strategies to safeguard FL systems from malicious attacks while preserving scalability, model performance, and data privacy. By achieving this, the framework aims to enhance confidence in FL technologies and foster their wider adoption in privacy-sensitive domains, particularly in healthcare applications such as disease prediction.

The objectives include designing and implementing a defense mechanism against adversarial attacks in FL, implementing privacy-preserving mechanisms that balance security, privacy and model performance, assessing the framework's ability to detect and mitigate attacks while maintaining model accuracy in healthcare scenarios and analyzing scalability and efficiency as FL networks expand. The research questions are: How can we effectively detect and mitigate adversarial attacks in FL without negatively affecting data privacy or model utility? To what extent can the proposed framework detect and protect against adversarial attacks while maintaining model performance and scalability in real-world healthcare environments?

This study suggests a defense-enhanced FL architecture that protects data privacy and model performance from adversarial attacks to meet the urgent demand for secure FL systems in the healthcare industry. Our strategy incorporates sophisticated security features such as adversarial training, differential privacy and Byzantine-robust aggregation which have been verified using a six-phase technique on the Mayo Clinic PBC dataset.

The framework's robust attack detection (84.33% detection rate) and capacity to retain an accuracy of 85.97% under assault settings are demonstrated by experimental findings thus providing a workable solution for privacy-sensitive healthcare applications such as disease prediction. This research improves its dependability for practical implementation by filling a significant gap in secure FL.

The remainder of this study is organized as follows: Section II presents related work, while Section III outlines the proposed methodology. Section IV details the experimental setup, and Section V shares the results and discusses their implications. Finally, Section VI concludes with remarks and suggests future directions for research.

II. Related Work

Edge computing and FL are complementary technologies that aim to address distributed data processing and machine learning challenges. FL addresses privacy and regulatory concerns by enabling model training on dispersed datasets while allowing multiple parties to collaborate on model training while keeping their data localized. Participating devices receive a global model from a central server, which initializes and distributes it. Edge devices train the model using their local data and only communicate model updates to the server [11]. Edge computing, a distributed computing paradigm, moves data storage and processing closer to the data sources [12]. It improves real-time processing, saves bandwidth, and reduces latency. Since the network edge generates substantial volumes of data, edge computing is crucial to FL. Benefits include enhanced data security and privacy, optimized bandwidth, reduced latency, increased reliability in intermittent connectivity, and support for real-time applications and decision-making [13]. FL and edge computing support data privacy by storing sensitive data locally. Edge computing minimizes data transfer, thereby reducing communication overhead, while FL simply requires model updates [14]. Rapid scenario adaptation is made possible by edge devices, which do local training and inference [15]. Architectures like Wu et al.'s [16, 17] hierarchical edge-based FL eliminate communication bottlenecks and improve scalability. Peer-to-peer FL eliminates the central server, while hybrid edge-cloud FL combines cloud and edge computing resources.

Threats originate from clients, communication and servers in FL. Clients face various attacks, including data poisoning, model poisoning, backdoor attacks, Byzantine attacks, Sybil attacks, free-riding, and inference attacks. Vulnerabilities in communication often arise from man-in-the-middle attacks and eavesdropping, which compromise data integrity and confidentiality. The central server faces risks from malicious behaviors, non-robust aggregation methods, and inference attacks [18]. Model poisoning attacks involve malicious participants injecting updates to manipulate the global model. Bhagoji et al. [19] demonstrated that an adversary controlling a single agent can achieve targeted misclassification. These attacks are stealthy and bypass simple anomaly detection. Data poisoning exploits the fact that FL aggregators are unaware of how updates are generated. Demartis [20] showed that even a small number of malicious participants can harm the joint model. Backdoor attacks involve malicious clients embedding hidden patterns in their updates, causing the model to misbehave on specific inputs. Unlike data poisoning, backdoor attacks maintain high accuracy on normal data but only activate under specific conditions. This type of attack exploits FL's decentralized nature. The decentralized architecture of FL makes it challenging to detect malicious updates [18]. The central server has limited visibility into the data of clients and training processes [21]. Edge-based FL introduces security concerns, as edge servers protect edge traffic but can be compromised, potentially impacting connected clients or manipulating aggregated updates. Privacy concerns extend beyond the protection of raw data. Inference attacks, which utilize membership, attribute, and feature inference, can retrieve the original data from model changes [14, 22]. Byzantine-resilient aggregation, differential privacy, secure aggregation protocols and anomaly detection are some of the protection measures that researchers suggest.

FL employs various defense mechanisms to safeguard against security and privacy anomalies at the client, server, and communication levels. At the client level, techniques such as differential privacy and anomaly detection filter malicious updates before aggregation. On the server side, robust aggregation methods like Krum and multi-Krum mitigate the impact of poisoned data and prevent non-robust aggregation issues. In the event of malicious client behavior, Byzantine fault tolerance ensures model integrity. Secure channels protect against eavesdropping and man-in-the-middle attacks, while encryption and moving target defenses enhance data transmission security. Robust aggregation identifies and filters harmful client updates. According to Bhagoji et al. [19], Byzantine-resilient aggregation techniques safeguard against model poisoning attacks but may be vulnerable to highly skilled targeted attacks. These aggregation algorithms statistically analyze client updates to identify outliers or unusual patterns of activity. Differential privacy is a privacy-preserving technique that adds controlled noise to gradients or model updates to maintain individual privacy. Shaheen et al. [11] proposed a client-level differential privacy approach for FL that offers robust privacy assurances without compromising model utility. Edge-specific security solutions address challenges in edge computing environments. Bao et al. [14] proposed a hierarchical edge-based FL architecture with intermediate aggregation layers, reducing communication bottlenecks and enhancing scalability while improving security.

Technologies like FL and Edge Computing are revolutionizing the healthcare industry by addressing challenges related to data security, privacy, and collaborative research. FL utilizes diverse datasets to enhance performance by enabling multiple institutions to train machine learning models without sharing raw patient data. A systematic study conducted by Teo et al. [8] identified 612 articles exploring the application of FL in healthcare, with internal medicine and radiology emerging as the most prevalent specialties. Neural networks and medical imaging are two prevalent models and data types that FL can effectively handle. Notably, only 5.2% of the examined research demonstrated real-life applications, suggesting early clinical use despite the growing interest in this field [8]. FL provides privacy by localizing data, but additional privacy enhancement methods are being developed, such as differential privacy, homomorphic

encryption, and secure multi-party computation to protect against potential privacy breaches during model updates [23]. Kyung Hee University used FL to create a clinical decision support system based on deep learning, thus facilitating extensive data mining and helping medical personnel make precise diagnoses and treatment choices [23]. Drug discovery has also made use of FL; ten pharmaceutical companies and academic universities collaborated to build a big industry-scale FL model for drug discovery without disclosing private data. The combination of Edge Computing with FL improves healthcare AI systems by processing data locally on edge devices, hence lowering latency and decreasing data transmission. For effective privacy-preserving medical research and patient care, FL and edge computing are essential [24]. Differential privacy methods can be successfully applied to clinical and epidemiological research, reproducing diverse health studies in a federated setting while maintaining data privacy.

In the healthcare industry, FL and edge computing improve privacy, minimize latency and boost productivity. Nonetheless, managing communication overhead and computational resources are significant obstacles. Complex machine learning models and substantial processing power are needed for healthcare applications, but edge devices may not be able to meet these demands [8, 25]. Model compression and selective parameter updates are two optimization strategies that save computational load without sacrificing accuracy [8]. Frequent model updates result in communication overhead that raises latency and network traffic [9]. Particularly in large-scale healthcare systems with several devices and institutions hierarchical FL methods with intermediate aggregation nodes improve scalability and lower costs [23]. The performance of FL systems is challenged by data heterogeneity across healthcare devices and institutions. Model bias and decreased generalization result from differences in data distribution, format and quality [10]. Adaptive FL algorithms improve performance in healthcare applications such as medical image analysis and disease prediction by handling non-IID data and adjusting model updates according to local variables [8]. When FL and edge computing integrate with the existing healthcare infrastructure scalability problems arise. Outdated hardware and software may not be compatible with modern FL frameworks [9]. By adjusting to different healthcare scenarios and gradually adding edge computing capabilities, modular FL designs enable institutions to adopt FL and edge computing technologies at their own pace [23]. Security and privacy constraints significantly impact FL systems' performance and scalability. Although FL offers data privacy by default, extra precautions are needed to guard against attacks and breaches [10, 25]. Stronger privacy assurances are offered by privacy-enhancing strategies like secure multi-party computation and differential privacy, but these come with extra communication and computational costs that must be weighed against performance demands.

A. Research Limitations and Identified Gaps

While prior research has advanced the security and application of federated learning (FL), several limitations persist, underscoring gaps that this study addresses. Table I summarizes key limitations in existing work and how our proposed defense-enhanced FL framework overcomes them.

TABLE I. LIMITATIONS OF EXISTING RESEARCH AND GAPS

Existing Research	Methodology	Limitations and Research Gaps
Research paper [19]	Analyzes model poisoning through adversarial lens, focusing on single-agent attacks	Limited to single-agent model poisoning; lacks defenses for multi-agent attacks or diverse attack types like data poisoning and backdoors
Research paper [8]	Systematic review of FL applications in healthcare, analyzing 612 studies	Only 5.2% of FL healthcare studies demonstrate practical applications, indicating a gap in real-world implementation.
Research paper [32]	Employs Byzantine-robust aggregation for federated learning	Byzantine-robust aggregation alone is insufficient to counter data poisoning or backdoor attacks, limiting comprehensive security.
Research paper [36]	Investigates data poisoning in sequential and parallel FL settings	Narrow focus on sequential and parallel FL poisoning, overlooking other attack types like model poisoning and backdoors

III. METHODOLOGY

To achieve the first research objective, the proposed methodology employs adversarial attacks to analyze their impact on FL models for disease prediction. The framework will incorporate cutting-edge techniques such as homomorphic encryption, differential privacy, and adversarial training. The performance of the framework will be evaluated based on its ability to detect and thwart attacks while maintaining high model accuracy and data privacy. The FL environment will be established, and the outcomes of various defensive strategies will be compared to determine the most effective approach. Fig. 1 outlines the research framework.

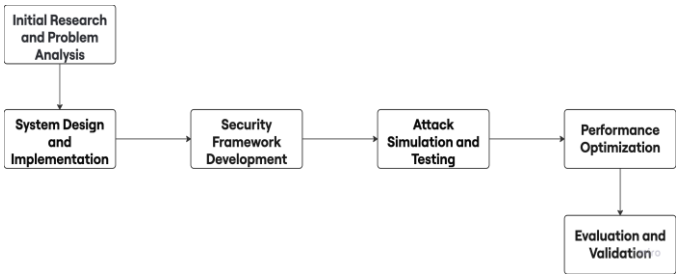

Fig. 1. Flowchart of research.

The study develops and validates a secure FL defensive framework for healthcare using a six-phase methodology. To identify critical vulnerabilities in the current defenses against risks such as model poisoning and data poisoning, the initial steps involve analyzing adversarial attack patterns and FL frameworks. This process guides the development of a two-tiered defense architecture that integrates server-side security features with client-side safeguards. To evaluate the effectiveness of detection, the system undergoes stress testing using attack scenarios on healthcare datasets. Following attacks, the system iteratively refines security, accuracy, and privacy. The final evaluation is assessed using metrics like attack detection rate, false positive rate, and accuracy. The

framework's practical applicability through encrypted communication is demonstrated through validation in a multi-institutional disease prediction scenario utilizing Kaggle data. Scalability among 2 to 20 healthcare nodes is ensured by ongoing performance monitoring thus maintaining the utility of the model.

A. Dataset Description and Preparation

The Mayo Clinic's 1974 to 1984 study on liver primary biliary cirrhosis (PBC) provided the dataset for this investigation. It was acquired from the UCI Machine Learning Repository and Kaggle [26]. The subject of this dataset is cirrhosis, a severe liver disease brought on by long-term damage caused by hepatitis or sustained alcohol use. The dataset includes attributes such as number of days between registration and the earlier of death, transplantation, or study analysis time, status, drug, age, sex, ascites, hepatomegaly, spiders, edema, bilirubin, cholesterol, albumin, copper, alkaline phosphatase levels, serum glutamic oxaloacetic transaminase levels, triglycerides, platelets, prothrombin and stage. The dataset comprises 25000 records and 19 features and is relevant for analyzing patient survival and disease progression patterns, making it suitable for machine learning models aimed at cirrhosis stage prediction.

There are several crucial elements in the dataset preparation process for FL. Categorical variables are one-hot encoded to ensure model compatibility, and missing values are eliminated to maintain data consistency. StandardScaler from scikit-learn is employed to standardize continuous variables, thereby enhancing model convergence. To adhere to PyTorch's CrossEntropyLoss specifications, the target variable "Stage" undergoes label encoding. Subsequently, the dataset is divided into 90% training and 10% testing sets. The training data is subsequently distributed among twenty clients for the FL setup. These procedures are carried out by the preprocessing function which guarantees that the dataset is clear and appropriate for machine learning model training in this configuration. The distribution of stage classes in the liver cirrhosis dataset reveals a nearly equal split across stages 1, 2, and 3. Stage 2 has the highest count (8441), followed closely by Stage 3 (8294) and Stage 1 (8265).

B. Core Algorithms

The core algorithms that form the basis of our FL system, both in its baseline configuration and with enhanced defense mechanisms are listed below.

1) Baseline FL algorithm: In FL, private data is utilized for on-device local training for each client, such as hospitals. For this multi-class problem of disease stage prediction, clients train using PyTorch's AdamW optimizer and CrossEntropyLoss, executing thirty epochs with a batch size of sixty-four to balance efficiency and learning. To safeguard privacy, model weights are independently created and transmitted to a central server for aggregation. The aggregation process on the server employs weighted averaging, as illustrated in Eq. (1), based on the size of the dataset, where clients with more data have greater influence.

$$\mathbf{w^{t+1}} = \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{w_k^{t+1}} \quad (1)$$

where, $\mathbf{w^{t+1}}$ is the global model's weight vector after aggregation, K is the number of clients, n_k is the number of samples for client k, n is the total number of samples across all clients, and $\mathbf{w_k^{t+1}}$ represents the local model weight vector from client k [27].

Uniform model architecture is assumed with zero padding for discrepancies. The global model is evaluated on a 10% test set using accuracy, precision, recall, and F1-score, which are averaged across classes, i.e., Stages 1, 2, 3. Early stopping halts training if test accuracy improvement drops below $\Delta\_min = 0.001$ over five rounds for efficiency and to prevent overfitting.

2) Defense-Enhanced FL algorithm: Clients perform local training utilizing differential privacy and adversarial training to protect against data leaks and adversarial assaults once the central server initializes and distributes a global model to clients. The global model is updated and checked for any attacks or performance degradation after model updates are safely aggregated using Byzantine-robust techniques to reduce malicious contributions. After that, a centralized test set is used to evaluate the updated model, and early stopping conditions are analyzed to decide whether to continue. To balance efficiency, security and model accuracy throughout the FL lifecycle, this cycle—local training, secure aggregation, verification, evaluation and stopping checks—repeats iteratively until convergence or a predetermined maximum number of rounds is reached. Algorithm 1 shows FL with early stopping.

Algorithm 1 FL with early stopping

```
INITIALIZE global model, defender, best_accuracy, rounds_without_improvement.
for each round (1 to max rounds):
    reset client models and client data sizes.
    for each client:
        validate client data
        train local model with differential privacy and adversarial robustness.
        validate local model
        encrypt and append valid models to client models.
        if defense enabled:
            aggregate models using defender.secure_aggregate.
        skip round if global model fails verification.
evaluate global model
    update best_accuracy if improvement > min_delta; else, increment rounds_without_improvement.
    stop if rounds_without_improvement >= patience.
return final global model
```

3) Secure aggregation: Secure aggregation integrates model updates from multiple clients while protecting individual privacy. The process involves several steps: the server decrypts encrypted model updates using an EncryptionSetup for secure decryption; it then aggregates the updates using trimmed mean aggregation, mitigating malicious updates or outliers; and

preserves the original model parameter shapes for compatibility with the global model architecture; finally, momentum stabilization smooths updates, enhancing convergence (see Algorithm 2).

Algorithm 2 Secure aggregation

```
1: FUNCTION secure_aggregate(global_model, client_models, client_data_sizes):
2:     initialize shapes registry if not already set
3:     Decrypt client models
4:     for each key in global model parameters:
5:       stack all client updates for this key
6:       sort updates
7:       compute trimmed mean by discarding extreme values
8:       update global model parameter with trimmed mean
9:     if best global model exists:
10:        apply momentum stabilization
11:  load updated parameters into global model
12:  return updated global model
```

4) Defense mechanisms: Different algorithms make machine learning systems more secure and resilient, especially in FL settings. While adversarial training strengthens model resilience by using adversarial cases during training differential privacy adds noise to model updates to protect individual privacy. Data validation identifies possible poisoning threats by evaluating data quality through tests for NaN values, outliers and label distribution, while Byzantine-robust aggregation uses trimmed mean aggregation to combat fraudulent updates from compromised clients. Model validation evaluates locally trained models against accuracy, loss, and consistency metrics to identify poisoning, whereas dynamic thresholding filters out suspicious updates using an adaptive interquartile range (IQR) approach. Model verification rollback monitor performance and restore it to a previous state if degradation is found, ensuring global model integrity. When combined, these techniques tackle the issues of integrity, resilience and privacy in distributed learning systems.

Differential Privacy protects individual privacy by adding controlled noise to model updates, as shown in the pseudo-code, where gradients are clipped to a specified norm (clip_norm) and Gaussian noise is added based on a noise_scale parameter. This ensures that the output from the model does not expose unique individual contributions by limiting the influence of any one data point (see Algorithm 3).

Algorithm 3 Differential Privacy

```
1: FUNCTION add_differential_privacy(model, clip_norm, noise_scale):
2:     total_norm = clip_gradients(model, clip_norm)
3:     for param in model.parameters():
4:       if param.grad is not null:
5:           noise = generate_gaussian_noise(param.grad.shape, scale=noise_scale)
6:       param.grad.add_(noise)
7:     return total_norm
```

By creating adversarial instances using the Fast Gradient Sign Method (FGSM), as shown in the pseudo-code, where inputs are disrupted by an epsilon-scaled gradient sign to maximize loss, Adversarial Training improves the robustness of the model. The model is then trained using these instances to increase its resistance to malicious disturbances (see Algorithm 4).

Algorithm 4 Adversarial Privacy

```
1: FUNCTION generate_adversarial_examples(model, loss_fn, x, y, epsilon):
2:     x_adv = x.clone().detach().requires_grad_(True)
3:     outputs = model(x_adv)
4:     loss = loss_fn(outputs, y)
5:     gradients = compute_gradients(loss, x_adv)
6:     x_adv = x_adv + epsilon * sign(gradients)
7:     x_adv = clip(x_adv, 0, 1)
8:     return x_adv.detach()
```

C. Initial FL System

The experimental architecture depicted in Fig. 2 comprises three crucial components: a central server responsible for initiating and updating the global model using the Federated Averaging (FedAvg) algorithm, clients representing healthcare institutions that train local models on their datasets and subsequently transmit updates to the server, and secure communication channels that facilitate the transmission of model updates between the server and clients.

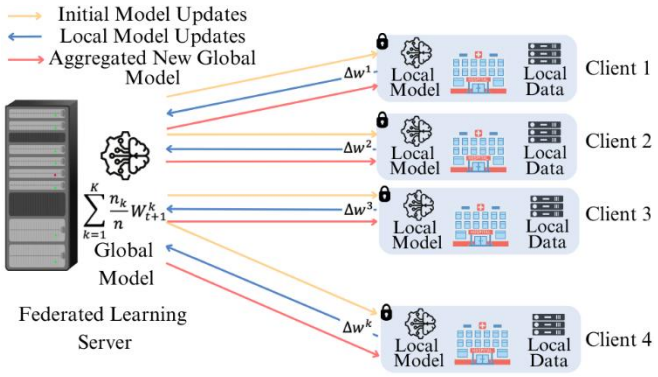

Fig. 2. Initial FL system setup.

The setup is an FL environment using a healthcare dataset. It consists of a central server and multiple clients, each with a local dataset. The central server manages the global model. It distributes an initial model to all clients, initiates local training, and collects model updates, i.e., weight and bias updates from clients. The server aggregates these updates to create an improved global model, redistributes it to clients and iteratively improves the model until the accuracy stops significantly improving when steady state is reached. This process is visualized with color-coded lines: orange for initial global model distribution, blue for local model updates and red for the aggregated global model distribution.

The neural network global model in Fig. 3 was designed for multi-class disease stage classification. It comprises an input layer that receives preprocessed feature vectors, followed by three fully connected hidden layers. Each hidden layer has 256,

128, and 64 neurons, respectively. These layers employ ReLU activation and dropout (rate 0.1) to enhance learning and mitigate overfitting. The output layer consists of three neurons and employs softmax activation to generate class probabilities. The model is appropriate for FL across a variety of computational resources since it makes use of CrossEntropyLoss, regularized by weight decay and optimized with AdamW.

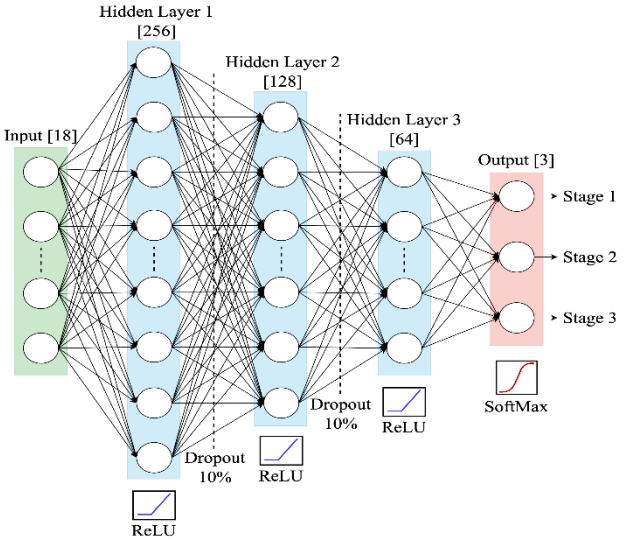

Fig. 3. Neural network architecture for Cirrhosis stage prediction.

The decentralized organization depicted in Fig. 4 is modeled by the FL training procedure. Each client employs PyTorch's AdamW optimizer and CrossEntropyLoss to train a local model on its dataset for thirty epochs, with a batch size of sixty-four. After training, clients transmit their model weights to the central server, which employs the weighted averaging technique Eq. (1) to aggregate them. The accuracy, precision, recall, and F1-score of the global model are evaluated using macro-averages across disease stages (1, 2, 3). To optimize efficiency and prevent overfitting, an early stopping mechanism terminates training if the test accuracy does not substantially increase (Δ_min=0.001) over five rounds.

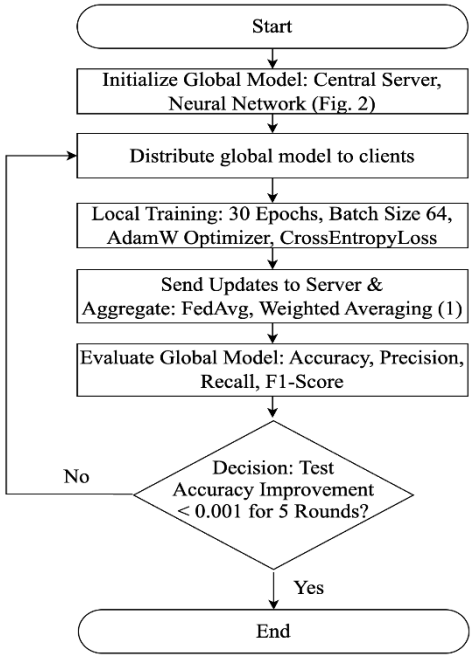

Fig. 4. FL System workflow.

D. Defense-Enhanced FL System

The Defense-Enhanced FL framework protects against adversarial threats while ensuring data privacy and maintaining the utility of the model (see Fig. 5).

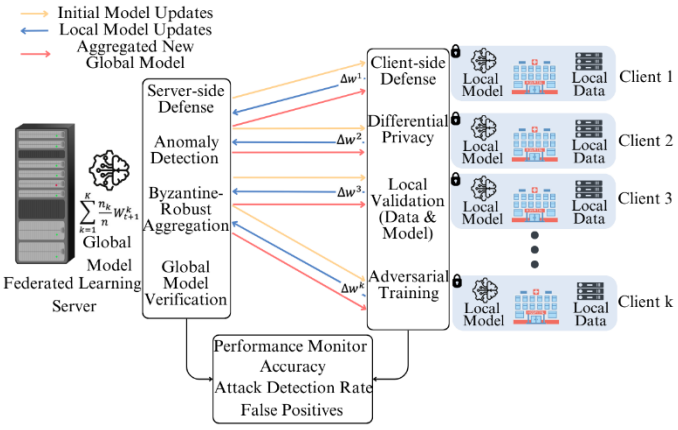

Fig. 5. FL System defense framework.

Server-side defenses employ anomaly detection, robust aggregation, and global model verification to safeguard against adversarial threats. In contrast, client-side defenses utilize differential privacy, adversarial training, and local validation to guarantee secure contributions to the global model.

1) Anomaly detection. The server utilizes robust z-score calculation and dynamic thresholding techniques to identify and eliminate outliers. Robust z-score is achieved using Eq. (2) and dynamic thresholding with Eq. (3).

$$Z_i = \frac{|x_i - \tilde{x}|}{MAD + \epsilon} \quad (2)$$

where, x_i represents the parameter values, $\tilde{x}$ is the median, MAD is the median absolute deviation, and ϵ is a small constant to prevent division by zero [29].

$$\text{Upper Bound} = Q_3 + \text{Sensitivity} \times \text{IQR} \quad (3)$$

Here, Q_3 is the third quartile and Sensitivity controls the threshold's strictness [30].

2) Byzantine-robust aggregation. Trimmed Mean Aggregation removes extreme values from client updates before averaging to minimize the impact of outliers, as illustrated in Eq. (4). Momentum Stabilization merges the current global model with historical models to enhance robustness, as per Eq. (5).

$$\theta_{global} = \frac{1}{|S|}\sum_{i \epsilon S} \theta_i \quad (4)$$

where, θ_{global} is the gobal model, θ_i is a client model and S represents the set of trimmed client updates after removing a percentage of extreme values based on the trim ratio [31].

$$\theta_{stabilized} = (1-\alpha)\cdot\theta_{current} + \alpha\cdot\theta_{historical} \quad (5)$$

Here, α controls the influence of past models on the current update [32].

3) Global model verification. The server continuously validates the global model's quality using validation datasets. If the accuracy drops significantly, a rollback mechanism automatically restores the previously validated model state.

4) Differential privacy. This ensures that individual data points are not inferred from model updates by adding noise to gradients during local training. This is accomplished through gradient clipping using Eq. (6) and noise addition using Eq. (7), which strikes a balance between privacy and model accuracy.

$$g' = \frac{g}{\max\left(1, \frac{||g||_2}{C}\right)} \quad (6)$$

where, g is the gradient vector and C is the clipping norm.

$$g'' = g' + N(0, \sigma^2) \quad (7)$$

Here, N is a Gaussian distribution, σ controls the noise scale, balancing privacy and model accuracy.

5) Adversarial training. This approach exposes the model to adversarial examples during local training, enhancing its resilience to evasion attacks without compromising performance on clean data, as demonstrated in Eq. (8).

$$x_{adv} = x + \epsilon \cdot \text{sign}\left(\nabla_x L(f(x;\theta), y)\right) \quad (8)$$

where, x_{adv} is the adversarial example, x is the original input, y is the label, $f(x;\theta)$ is the model prediction, L is the loss function and ϵ controls perturbation magnitude.

6) Client data validation. Clients validate local datasets for anomalies and poisoning attempts before training. This ensures that the local models are not corrupted. Outlier detection and label distribution checks are performed to achieve this [Eq. (9)].

$$Q1 - k \cdot IQR < x < Q3 + k \cdot IQR \quad (9)$$

where, Q1 and Q3 are the first and third quartiles, respectively, IQR = Q3-Q1 and k for strict filtering.

7) Local model verification. Clients validate trained models using validation data to ensure minimum accuracy, consistency, and robustness against adversarial inputs as seen in Eq. (10).

$$C_{adv} = \frac{\sum_{i=1}^{N}\left(\hat{y}_i == \hat{y}_{adv,i}\right)}{N} \quad (10)$$

where, C_{adv} is adversarial accuracy, N is the total samples, $\hat{y}_i$ is the predicted label for the original input and $\hat{y}_{adv,i}$ is the predicted label for adversarial input.

8) Communication encryption and secure aggregation. Additive noise encryption is employed to establish secure communication between clients and the server. Encrypted updates are aggregated from multiple clients without revealing individual contributions. This ensures that even if an adversary gains access to updates on the server side, they cannot reconstruct individual updates due to the added noise.

E. Attack Setup

Data poisoning attacks corrupt training data to manipulate a model's behavior, posing a unique threat in federated learning (FL) due to malicious clients lacking direct access to the central model. In this framework, a function employs a label flipping technique. This technique involves changing a predetermined percentage of labels (determined by the poison ratio parameter) to false values using a simple increment with modulo operation and a random selection procedure. The altered dataset is then returned with an "is_malicious" flag to mimic detection mechanisms, while the randomization aids in avoiding detection. These attacks have serious repercussions, as they can lower model accuracy, produce inaccurate data associations, and even open backdoors for certain misclassifications.

Model poisoning attacks target the integrity of FL by altering model updates from malicious clients, directly affecting the aggregation process. The framework's implementation involves adding random noise to model parameters, controlled by the attack_strength parameter, which adjusts the perturbation's intensity. This ensures that the poisoned model remains structurally compatible with the system. Similar to data poisoning, model poisoning attacks include an "is_malicious" flag for detection. These attacks can severely impair the global model's performance, introduce hard-to-detect backdoors or biases, and potentially cause targeted misclassifications, making them formidable challenges in FL environments.

Backdoor attacks aim to embed hidden triggers in the global model, causing misclassifications only when specific patterns are present while preserving accuracy on normal data. To mimic this behavior, the framework reassigns a target label to a subset of training data that has a trigger pattern added to it, as specified by the backdoor_ratio parameter. The function returns the modified dataset with an "is_malicious" flag, ensuring that the subtle yet reliable trigger remains concealed. These attacks pose a significant risk because they can activate under specific conditions undetected, leading to persistent vulnerabilities that are challenging to identify or eliminate, even with additional training.

Man-in-the-middle (MITM) attacks threaten FL by intercepting and modifying communications between clients and the server, which is set up in the framework to test system resilience. In addition to handling both encrypted and unencrypted arguments while maintaining system compatibility, the attack function incorporates an "is_malicious" flag and modifies model updates by introducing noise scaled by attack_strength. MITM attacks highlight the importance of robust security measures in FL systems, as they can gradually degrade the global model, compromise process integrity, and potentially enable model poisoning or backdoor insertion through persistent update manipulation.

F. Evaluation Metrics

1) FL model performance metrics. The performance of the FL system is evaluated using the following four key metrics:

a) Accuracy: The overall correctness of the model's predictions, which is calculated as the ratio of correctly classified instances to the total number of instances [28] [see Eq. (11)]:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (11)$$

where, TP stands for True Positives, TN for True Negatives, FP for False Positives and FN for False Negatives.

b) Precision: This evaluates the proportion of correctly predicted positive cases out of all predicted positive cases. It is particularly useful in scenarios, where false positives are costly, [see Eq. (12)]:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (12)$$

High precision indicates that the model makes fewer false positive errors [28], which is critical in healthcare applications, such as disease prediction.

c) Recall (Sensitivity): Recall measures how many actual positive cases were correctly identified by the model. High recall ensures that most actual positive cases are detected [28], which is crucial for minimizing missed diagnoses in healthcare [see Eq. (13)].

$$\text{Recall} = \frac{TP}{TP + FN} \quad (13)$$

d) F1-Score: This is the harmonic mean of precision and recall, providing a single metric that balances both [see Eq. (14)].

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times Recall}{\text{Precision} + \text{Recall}} \quad (14)$$

2) Defense framework performance metrics. Key performance metrics of the defense framework across security, latency and scalability are tracked by the performance monitoring system as follows:

a) Attack Detection Rate (True Positive Rate): Measures the proportion of actual attacks correctly identified by the system out of all attacks [Eq. (15)].

$$\text{Attack Detection Rate} = \frac{TP}{TP + FN} \quad (15)$$

b) False Positive Rate (FPR): Evaluates the proportion of benign updates incorrectly flagged as attacks [Eq. (16)].

$$\text{FPR} = \frac{FP}{FP + TN} \quad (16)$$

c) Precision: Assesses the accuracy of attack detection by calculating the proportion of flagged updates that are truly malicious.

d) Latency metrics: Aggregation latency refers to the time the server takes to combine client updates into a global model, while validation latency measures the duration needed to validate this global model against a reference dataset after aggregation. These processes together contribute to the average round time which encompasses the total time required for one complete cycle of communication, training, aggregation and validation [Eq. (17)].

$$\text{Average Metric} = \frac{\sum_{i=1}^{N} \text{Latency}_i}{N} \quad (17)$$

where, *N* is the number of rounds completed. By monitoring these latencies, the system measures the computational overhead that defense mechanisms introduce.

IV. Experimental Setup

A MacBook M3 system with an 8-core Apple M3 CPU, 16GB of unified RAM, a 1TB SSD and macOS Sequoia (version 15.1) forms part of the hardware setup. This configuration offered sufficient processing capacity for running adversarial attack setups, FL scenarios, and training medium-sized machine learning models. PyTorch computations were optimized by the M3 chip's sophisticated architecture, especially for gradient updates and encryption jobs, thereby guaranteeing effective performance throughout the tests.

The software environment was built around Python 3.12.4 as the primary programming language supported by a suite of development tools and libraries tailored for machine learning. iTerm2 oversaw the execution of FL code, while Jupyter Notebook enabled interactive prototyping and visualization, and Visual Studio Code functioned as the primary IDE, supplemented by extensions such as Python and Jupyter. Important libraries included NumPy and Pandas for data manipulation, scikit-learn for preprocessing and evaluation, matplotlib and seaborn for visualizing performance metrics and data trends and PyTorch for building and training neural networks with GPU acceleration via Metal Performance Shaders. The FL system and its defense mechanisms may be implemented, trained and evaluated thanks to this all-inclusive environment. GitHub and Git were utilized for collaboration and version control.

V. Results and Discussion

A. Feature Correlations

The heatmap shows moderate relationships between biomarkers, suggesting interdependent physiological processes that federated ML can leverage.

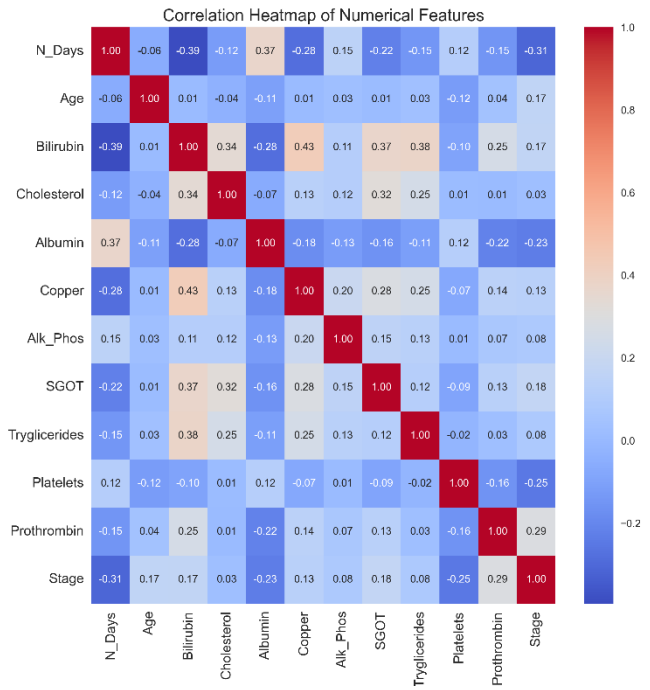

Fig. 6. Correlation matrix of features.

1) Baseline FL system model performance. Under benign conditions, the results reveal consistent performance over three test runs (Fig. 8). Precision, recall and F1-score average between 0.9007 and 0.9008, whereas the overall accuracy average is 90.07% (Table II). Plotting the federation rounds against accuracy shows a consistent upward trend, settling close to 90% for every run. While Stage 2 performs marginally worse (89.27%), Stage 3 attains the best accuracy (91.66%) and recall (0.9166). With AUC values ranging from 0.95 to 0.96, the ROC curve (Fig. 9) verifies strong bias for every class. High diagonal values indicate strong true positive rates, although overall classification is accurate, with the confusion matrix highlighting misclassifications between adjacent stages (Fig. 7).

TABLE II. BASELINE FL CLASSIFICATION METRICS

Metric	Test run 1	Test run 2	Test run 3	Average
Accuracy	89.20%	90.60%	90.40%	90.07%
Precision	0.8920	0.9061	0.9044	0.9008
Recall	0.8920	0.9060	0.9040	0.9007
F1-Score	0.8920	0.9060	0.9041	0.9007

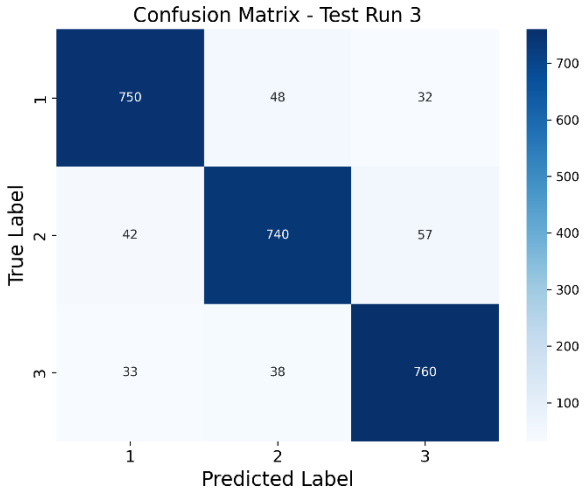

Fig. 7. Baseline FL model confusion matrix.

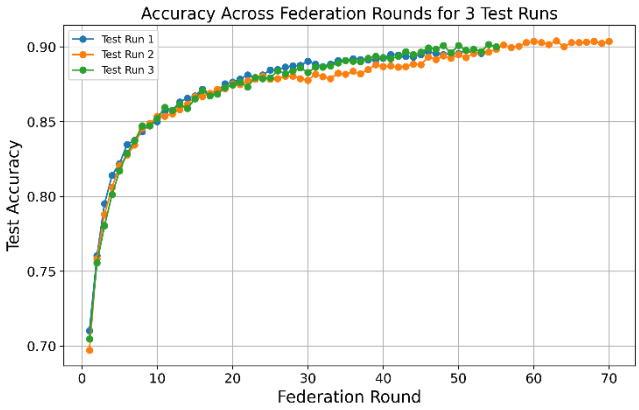

Fig. 8. Baseline FL model accuracy trend per test run.

2) Defense-enhanced FL system model performance. The defense-enhanced FL system demonstrated consistent performance for three test cycles, averaging 85.97% accuracy (Table III). The per-class measures (Fig. 11) showcased strong performance, with Stage 3 achieving the highest average accuracy of 87.97%. The ROC curves (Fig. 12) further demonstrated the system's classification ability, with AUC values of 0.93 for Classes 0 and 1 and 0.94 for Class 2. The confusion matrix (Fig. 10) indicated a balanced prediction with minimal misclassifications. After forty rounds, the accuracy trends exhibited a consistent improvement, stabilizing over 85%, indicating the system's convergence and dependability.

3) Defense framework performance. During adversarial setups, the defense framework demonstrated exceptional threat recognition capabilities. Over three test runs, it achieved a noteworthy precision of 99.32%, an accuracy of 96.62%, a low false positive rate of 0.15%, and an impressive average attack detection rate of 84.33% (as depicted in Table IV). Test Run 2's confusion matrix showcased excellent classification, with minimal instances of false positives and negatives (illustrated in Fig. 13). Moreover, the ROC curve, with an AUC of 0.96, effectively demonstrated strong discrimination (Fig. 14).

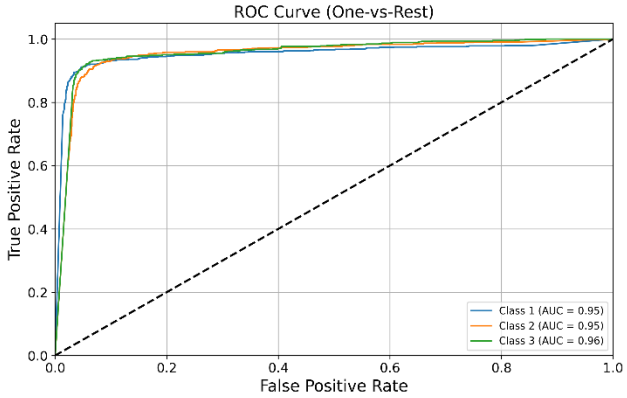

Fig. 9. Baseline FL model ROC curve.

TABLE III. DEFENSE-ENHANCED FL CLASSIFICATION METRICS

Metric	Test run 1	Test run 2	Test run 3	Average
Accuracy	86.88%	85.72%	85.32%	85.97%
Precision	0.8688	0.8548	0.8533	0.8590
Recall	0.8689	0.8572	0.8533	0.8598
F1-Score	0.8687	0.8572	0.8532	0.8597

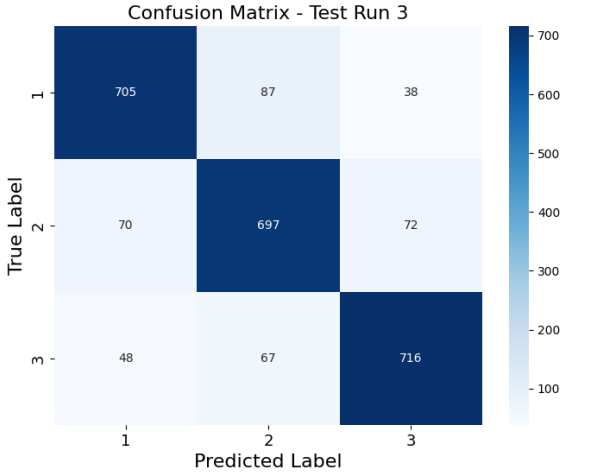

Fig. 10. Defense-enhanced FL model confusion matrix.

Fig. 11. Defense-enhanced FL model accuracy per test run.

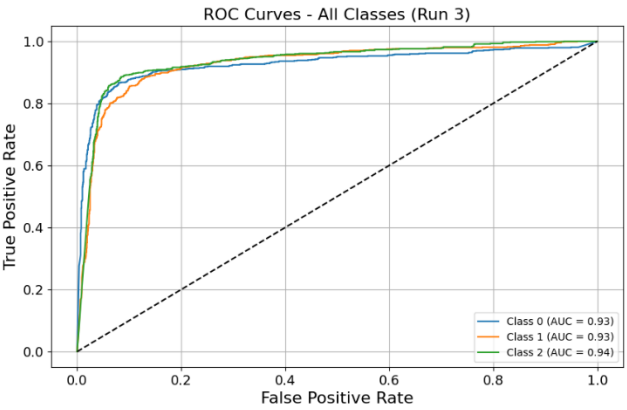

Fig. 12. Defense-enhanced FL model ROC curve.

TABLE IV. DEFENSE FRAMEWORK PERFORMANCE

Metric	Test run 1	Test run 2	Test run 3	Average
Attack Detection Rate	85.99%	84.26%	82.74%	84.33%
False Positive Rate	0.10%	0.00%	0.36%	0.15%
Precision	99.55%	100%	98.42%	99.32%
Accuracy	97.10%	96.72%	96.03%	96.62%

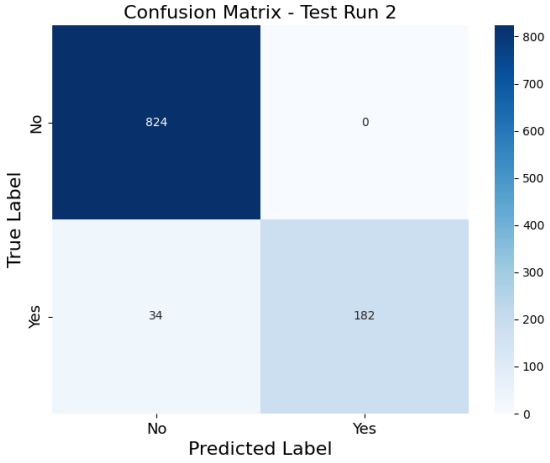

Fig. 13. Defense framework confusion matrix.

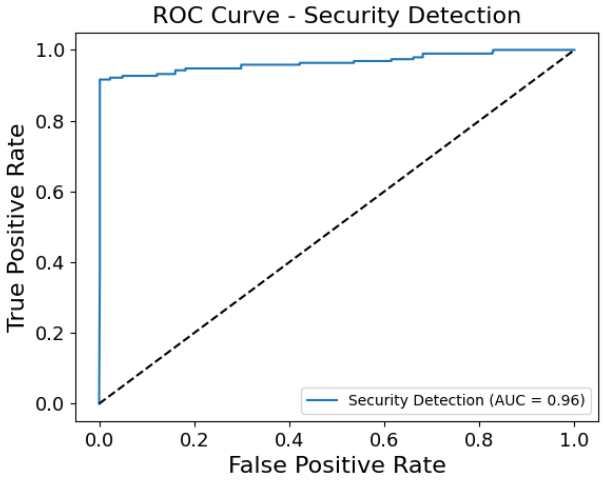

Fig. 14. Defense framework ROC curve.

4) Latency and scalability metrics. While Average Round Time stays constant at about 13 seconds with the baseline FL system (Fig. 15), Aggregation and Validation Latencies reduce marginally throughout the experiments. Average Round Time stays constant at 12 to 13 seconds, Validation Latency varies slightly but stays within a small range, and Aggregation Latency steadily rises as the number of customers rises from 2 to 20 (Fig. 16).

Aggregation Latency (~13.5–13.9 seconds), Validation Latency (~0.0009 seconds) and Average Round Time (~54–56 seconds) all exhibit consistency with the defense-enhanced FL model (Fig. 17). While Validation Latency constantly declines, Aggregation Latency rises with more clients, reaching a peak of 23.49 seconds for 14 clients before stabilizing. The average round time fluctuates, reaching a peak of 91.41 seconds for fourteen clients and then leveling off around 54 to 55 seconds for more clients (Fig. 18).

Fig. 15. Baseline FL system average aggregation, validation latency and round time for three test runs.

5) Comparative analysis. Performance, resilience against hostile attacks and effectiveness in healthcare applications are the main points of comparison between the initial FL system and the defense-enhanced FL system in this section. The defense-enhanced system's accuracy decreased to 85.97% (scores 0.8590–0.8598), a 4.1% decrease due to defense-related overhead, but it maintained reliable classification. The original FL system achieved an average accuracy of 90.07% with precision, recall and F1-scores around 0.9007–0.9008 on average. The defense-enhanced system showed remarkable resilience, improving security that is essential for healthcare settings, while the original system, which lacked defenses, is thought to be susceptible to hostile threats. Due to the additional computing load, efficiency favored the original system with round times of thirteen seconds as opposed to the defense-enhanced system's 54 to 56 seconds. The analysis identifies a trade-off: the defense-enhanced system forgoes some utility in favor of strong security, making it more appropriate for privacy-sensitive, real-world healthcare scenarios, whereas the original system excels in accuracy and speed under benign settings (see Table V).

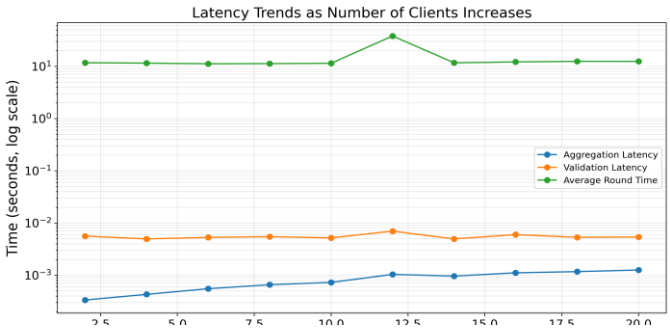

Fig. 16. Baseline FL system latency trends per increase in clients count.

Fig. 17. Defense enhanced FL system average aggregation, validation latency and round time for three test runs.

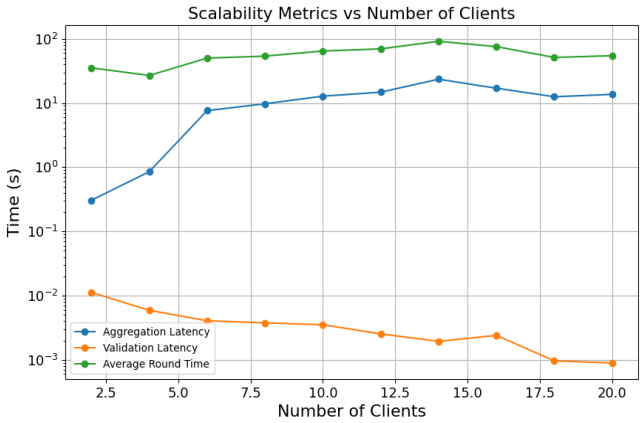

Fig. 18. Defense-enhanced FL system latency trends per increase in clients count.

TABLE V. COMPARATIVE ANALYSIS WITH EXISTING FL APPROACHES

Comparison Criteria	Original FL System (Baseline)	Defense-Enhanced FL System (Proposed Approach)	Existing Research on Secure FL
Accuracy (%)	90.07%	85.97% (↓ 4.1% due to defense overhead)	Varies (84–89%) [33] [34]
Precision / Recall / F1-score	0.9007 – 0.9008	0.8590 – 0.8598	Varies (0.80 – 0.85) [33] [35]
Resilience to Model Poisoning	Highly vulnerable	Strong protection (Byzantine-robust	Limited defenses (Most use secure aggregation only) [32] [42]

		aggregation, etc.)	
Resilience to Data Poisoning	No protection	Mitigated via anomaly detection	Partially addressed in some works [36] [37]
Resilience to Backdoor Attacks	Susceptible	Significantly reduced via secure model updates	Few studies implement full protection [36] [37]
Computational Efficiency (Training Round Time)	13 seconds	54–56 seconds (300%↑ due to security overhead)	Varies (~200% -~400%, depending on security measures used) [38] [39]
Scalability	High (Fast processing, limited security constraints)	Moderate (Additional security steps slow down processing)	Varies (Most methods struggle with large-scale deployment) [33] [40]
Suitability for Healthcare Applications	Vulnerable to attacks, making it risky for sensitive data	Highly secure, ensuring compliance with privacy laws (HIPAA, GDPR)	Most methods focus on general FL, not healthcare-specific defenses [2] [41]
Trade-offs	High accuracy & speed but weak security	Lower accuracy & speed but strong security	Varies (Many focus on either security or performance, not both) [42]

The performance of the proposed defense-enhanced FL framework, achieving an average accuracy of 85.97% on the Mayo Clinic PBC dataset, reflects its suitability for structured healthcare data with moderate feature correlations, as evidenced by the heatmap in Fig. 6. This dataset's balanced class distribution (Stage 1: 8265, Stage 2: 8441, Stage 3: 8294) and interdependent physiological features enable the framework to effectively leverage local training and aggregation. Variations in performance across different datasets, as seen in existing research (e.g., 84–89% accuracy in [33], [34]), likely stem from differences in data characteristics, such as class imbalance, noise levels, or feature correlations. The proposed algorithms excel with structured medical data exhibiting moderate to strong feature relationships, where the model can generalize across clients. However, on datasets with extreme imbalances or weak correlations—common in unstructured or heterogeneous healthcare data—performance may decline unless supplemented with preprocessing or adaptive techniques. This suggests that the framework's optimal application lies in well-structured, privacy-sensitive healthcare scenarios, with potential adaptations needed for noisier or less correlated data types.

VI. Conclusion and Future Work

This research successfully developed and validated a defense-enhanced federated learning (FL) framework tailored for privacy-sensitive healthcare applications, achieving its goal of enhancing security while maintaining model utility. By integrating differential privacy, adversarial training, and Byzantine-robust aggregation, the framework demonstrated robust protection against adversarial attacks, including data poisoning, model poisoning, and backdoors, with an attack detection rate of 84.33% and precision of 99.32%. Applied to the Mayo Clinic PBC dataset in a multi-institutional disease prediction scenario, it maintained an accuracy of 85.97%, despite a 4.1% drop due to security overhead, ensuring reliable classification (precision, recall, F1-scores ~0.8590–0.8598). The framework's latency stabilized at 54 to 56 seconds per round, reflecting a trade-off for enhanced security, making it a practical solution for healthcare settings compliant with privacy regulations like HIPAA and GDPR. These achievements establish a secure, scalable FL system that fosters trust in collaborative machine learning for sensitive domains. Future work will focus on reducing latency through hierarchical aggregation or gradient compression, validating the framework across diverse healthcare datasets like MIMIC-IV for broader applicability, and deploying it in real-world healthcare facilities to confirm its practical utility.

Acknowledgment

The authors gratefully acknowledge Norfolk State University, USA, for making the resources available. This material is based upon work supported by the National Science Foundation under Grant No. 2221099 and the U.S. Department of Energy's Office of Science (SC) under Award Number DE-SC0025722.

References

[1] K. Zhang et al., "FLIP: A provable defense framework for backdoor mitigation in federated learning." 2023. [Online]. Available: https://arxiv.org/abs/2210.12873.

[2] Y. Li, Z. Guo, N. Yang, H. Chen, D. Yuan, and W. Ding, "Threats and defenses in federated learning life cycle: a comprehensive survey and challenges." 2024. [Online]. Available: https://arxiv.org/abs/2407.06754.

[3] P. Liu, X. Xu, and W. Wang, "Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives," Cybersecurity, vol. 5, no. 1, p. 4, 2022.

[4] S. Lu, R. Li, W. Liu, and X. Chen, "Defense against backdoor attack in federated learning," Computers & Security, vol. 121, p. 102819, 2022, doi: https://doi.org/10.1016/j.cose.2022.102819.

[5] W. Wan, J. Lu, S. Hu, L. Y. Zhang and X. Pei, "Shielding federated learning: a new attack approach and its defense," 2021 IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 2021, pp. 1-7, doi: 10.1109/WCNC49053.2021.9417334.

[6] A. Shabbir, H. U. Manzoor, K. Arshad, K. Assaleh, Z. Halim, and A. Zoha, "Sustainable and lightweight defense framework for resource constraint federated learning assisted smart grids against adversarial attacks," Authorea Preprints, 2024, unpublished.

[7] X. Zhang, Y. Kang, K. Chen, L. Fan, and Q. Yang, "Trading off privacy, utility, and efficiency in federated learning," ACM Transactions on Intelligent Systems and Technology, vol. 14, no. 6, pp. 1–32, 2023.

[8] Z. L. Teo et al., "Federated machine learning in healthcare: a systematic review on clinical applications and technical architecture," Cell Reports Medicine, p. 101419, Feb. 2024, doi: https://doi.org/10.1016/j.xcrm.2024.101419.

[9] F. Zhang et al., "Recent methodological advances in federated learning for healthcare," Patterns, vol. 5, no. 6, 2024.

[10] M. S. Ali et al., "Federated learning in healthcare: model misconducts, security, challenges, applications, and future research directions–a systematic review," arXiv preprint arXiv:2405.13832, 2024.

[11] M. Shaheen, M. S. Farooq, and T. Umer, "AI-empowered mobile edge computing: inducing balanced federated learning strategy over edge for balanced data and optimized computation cost," Journal of Cloud Computing, vol. 13, no. 1, p. 52, 2024.

[12] H. G. Abreha, M. Hayajneh, and M. A. Serhani, "Federated learning in edge computing: a systematic survey," Sensors, vol. 22, no. 2, p. 450, 2022.

[13] Y. Qi, Y. Feng, X. Wang, H. Li, and J. Tian, "Leveraging federated learning and edge computing for recommendation systems within cloud computing networks." 2024. [Online]. Available: https://arxiv.org/abs/2403.03165.

[14] G. Bao and P. Guo, "Federated learning in cloud-edge collaborative architecture: key technologies, applications and challenges," Journal of Cloud Computing, vol. 11, no. 1, p. 94, 2022.

[15] X. Liu, X. Dong, N. Jia, and W. Zhao, "Federated learning-oriented edge computing framework for the IIoT" Sensors, vol. 24, no. 13, p. 4182, 2024.

[16] J. Wu, F. Dong, H. Leung, Z. Zhu, J. Zhou, and S. Drew, "Topology-aware federated learning in edge computing: a comprehensive survey" ACM Computing Surveys, vol. 56, no. 10, pp. 1–41, 2024.

[17] A. Brecko, E. Kajati, J. Koziorek, and I. Zolotova, "Federated learning for edge computing: a survey" Applied Sciences, vol. 12, no. 18, p. 9124, 2022.

[18] C. Zhang, S. Yang, L. Mao, and H. Ning, "Anomaly detection and defense techniques in federated learning: a comprehensive review," Artificial Intelligence Review, vol. 57, no. 6, pp. 1–34, 2024.

[19] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in International conference on machine learning, 2019, pp. 634–643.

[20] M. Demartis, "Adversarial attacks in federated learning," Dissertation, 2022, unpublished.

[21] J. Zhang et al., "Delving into the adversarial robustness of federated learning," in Proceedings of the AAAI conference on artificial intelligence, 2023, vol. 37, no. 9, pp. 11245–11253.

[22] K. N. Kumar, C. K. Mohan, and L. R. Cenkeramaddi, "The impact of adversarial attacks on federated learning: a survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.

[23] X. Gu, F. Sabrina, Z. Fan, and S. Sohail, "A review of privacy enhancement methods for federated learning in healthcare systems," International Journal of Environmental Research and Public Health, vol. 20, no. 15, p. 6539, 2023.

[24] C. S. Kruse, R. Goswamy, Y. J. Raval, and S. Marawi, "Challenges and opportunities of big data in health care: a systematic review," JMIR medical informatics, vol. 4, no. 4, p. e5359, 2016.

[25] W. Oh and G. N. Nadkarni, "Federated learning in health care using structured medical data," Advances in kidney disease and health, vol. 30, no. 1, pp. 4–16, 2023.

[26] Aadarsh velu, "Liver cirrhosis stage classification ," Kaggle.com, 2023. https://www.kaggle.com/datasets/aadarshvelu/liver-cirrhosis-stage-classification/data.

[27] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS), 2017, pp. 1273–1282.

[28] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," Information processing & management, vol. 45, no. 4, pp. 427–437, 2009.

[29] Y. Kim, H. Chen, and F. Koushanfar, "Backdoor defense in federated learning using differential testing and outlier detection," arXiv preprint arXiv:2202.11196, 2022.

[30] Ch. S. K. Dash, A. K. Behera, S. Dehuri, and A. Ghosh, "An outliers detection and elimination framework in classification task of data mining," Decision Analytics Journal, vol. 6, p. 100164, 2023, doi: https://doi.org/10.1016/j.dajour.2023.100164.

[31] T. Wang, Z. Zheng, and F. Lin, "Federated learning framework based on trimmed mean aggregation rules," Expert Systems with Applications, vol. 270, p. 126354, 2025, doi: https://doi.org/10.1016/j.eswa.2024.126354.

[32] K. Pillutla, S. M. Kakade and Z. Harchaoui, "Robust aggregation for federated learning," in IEEE Transactions on Signal Processing, vol. 70, pp. 1142-1154, 2022, doi: 10.1109/TSP.2022.3153135.

[33] K. Wei et al., "Federated learning with differential privacy: algorithms and performance analysis," IEEE transactions on information forensics and security, vol. 15, pp. 3454–3469, 2020.

[34] D. Stripelis et al., "A federated learning architecture for secure and private neuroimaging analysis," Patterns, vol. 5, no. 8, 2024.

[35] N. N. Albogami, "Intelligent deep federated learning model for enhancing security in internet of things enabled edge computing environment," Scientific Reports, vol. 15, no. 1, p. 4041, 2025.

[36] F. Nuding and R. Mayer, "Data Poisoning in Sequential and Parallel Federated Learning," in Proceedings of the 2022 ACM on International Workshop on Security and Privacy Analytics, 2022, pp. 24–34. doi: 10.1145/3510548.3519372.

[37] Y. Wan, Y. Qu, W. Ni, Y. Xiang, L. Gao, and E. Hossain, "Data and model poisoning backdoor attacks on wireless federated learning, and the defense mechanisms: a comprehensive survey," IEEE Communications Surveys & Tutorials, vol. 26, no. 3, pp. 1861–1897, 2024.

[38] K. Peng, X. Shen, L. Gao, B. Wang, and Y. Lu, "Communication-efficient and privacy-preserving verifiable aggregation for federated learning," Entropy, vol. 25, no. 8, p. 1125, 2023.

[39] K. Daly, H. Eichner, P. Kairouz, H. B. McMahan, D. Ramage, and Z. Xu, "Federated learning in practice: reflections and projections," in 2024 IEEE 6th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA), 2024, pp. 148–156.

[40] Z. Guan, Y. Zhao, Z. Wan, and J. Han, "OPSA: Efficient and verifiable one-pass secure aggregation with TEE for federated learning," Cryptology ePrint Archive, 2024.

[41] U. Zafar, A. Teixeira, and S. Toor, "Robust federated learning against poisoning attacks: a gan-based defense framework," arXiv preprint arXiv:2503.20884, 2025.

[42] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: a client-level perspective," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 5, pp. 1281–1292, May 2019.