FedHAN: A Cache-Based Semi-Asynchronous Federated Learning Framework Defending Against Poisoning Attacks in Heterogeneous Clients

Xiaoding Wang[1], **Bin Ye**[1], **Li Xu**[1], **Lizhao Wu**[1], **Sun-Yuan Hsieh**[2], **Jie Wu**[3,4] and **Limei Lin**[1*]
[1]College of Computer and Cyber Security, Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou 350117, China
[2]Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 701, Taiwan
[3]China Telecom Cloud Computing Research Institute, Beijing 100088, China
[4]Department of Computer and Information Sciences, Temple University, PA 19122, USA
wangdin1982@fjnu.edu.cn, yebin7048@gmail.com, xuli@fjnu.edu.cn, melowlz@yeah.net
hsiehsy@mail.ncku.edu.tw, jiewu@temple.edu, linlimei@fjnu.edu.cn

Abstract

Federated learning is vulnerable to model poisoning attacks in which malicious participants compromise the global model by altering the model updates. Current defense strategies are divided into three types: aggregation-based methods, validation dataset-based methods, and update distance-based methods. However, these techniques often neglect the challenges posed by device heterogeneity and asynchronous communication. Even upon identifying malicious clients, the global model may already be significantly damaged, requiring effective recovery strategies to reduce the attacker's impact. Current recovery methods, which are based on historical update records, are limited in environments with device heterogeneity and asynchronous communication. To address these problems, we introduce FedHAN, a reliable federated learning algorithm designed for asynchronous communication and device heterogeneity. FedHAN customizes sparse models, uses historical client updates to impute missing parameters in sparse updates, dynamically assigns adaptive weights, and combines update deviation detection with update prediction-based model recovery. Theoretical analysis indicates that FedHAN achieves favorable convergence despite unbounded staleness and effectively discriminates between benign and malicious clients. Experiments reveal that FedHAN, compared to leading methods, increases the accuracy of the model by 7.86%, improves the detection accuracy of poisoning attacks by 12%, and enhances the recovery accuracy by 7.26%. As evidenced by these results, FedHAN exhibits enhanced reliability and robustness in intricate and dynamic federated learning scenarios.

*Corresponding author

1 Introduction

Federated learning (FL) [Konečný *et al.*, 2017], a widely-used framework for distributed machine learning, is a significant research focus. Most FL algorithms, such as the classic FedAvg, fall into Synchronous Federated Learning (SFL). They require the server to wait for all selected clients' local training and uploads before aggregating updates, and assume uniform model sizes among clients. However, in reality, mobile devices often face limits in computational and communication capacities, which makes large models challenging. This results in extended training durations and slows FL progress. Consequently, semi-asynchronous federated learning algorithms that support model heterogeneity have become crucial [Sun *et al.*, 2023]. These methods assign models based on device capabilities and use a semi-asynchronous approach where the server aggregates the earliest received updates, bypassing the wait for all clients. This strategy better accommodates various devices and improves FL efficiency.

Due to its decentralized setup [Fung *et al.*, 2020; Bagdasaryan *et al.*, 2020; Fang *et al.*, 2020], federated learning is highly susceptible to poisoning attacks from malicious clients. Such clients might alter local data or manipulate model updates, leading to the degradation of the accuracy and reliability of the global model once these updates are incorporated. Current defenses are categorized into three main types: robust aggregation methods [Blanchard *et al.*, 2017; Chen *et al.*, 2017] that exclude suspicious updates based on statistical criteria but may also omit legitimate updates, reducing overall model accuracy; validation dataset-based methods [Cao *et al.*, 2021], which depend on the challenging task of creating representative validation datasets; and distance-based defenses [Zhang *et al.*, 2022; Huang *et al.*, 2023; Fung *et al.*, 2020; Xia *et al.*, 2019], using update distance measurements to detect malicious activities. Nevertheless, these defenses face further difficulties in practice due to model heterogeneity and asynchronous communication. In heterogeneous FL, the varying sizes and architectures of the client models lead to uneven update distributions, complicating the detection of malicious actions. In asynchronous FL, differing client upload times cause outdated updates that diverge from the latest global ones, disrupting synchronization and complicating the differentiation between benign and malicious updates.

In federated learning systems, standard approaches for detecting malicious clients generally depend on statistical analysis conducted after several attack rounds. Unfortunately, the global model might already be significantly compromised by the time these clients are recognized. Hence, immediate action is required once a malicious client is found. One solution involves adjusting model parameters and adding Gaussian noise [Xie *et al.*, 2021; Nguyen *et al.*, 2022], which can counteract backdoor attacks, but may reduce model efficiency. Alternatively, "machine unlearning" [Cao *et al.*, 2023; Liu *et al.*, 2021; Jiang *et al.*, 2025] can eliminate the effects of harmful updates, albeit at the cost of increased storage due to the need to retain past model updates. Ensuring the success of current recovery methods in real federated environments remains a challenge. To overcome these issues, we introduce a defense strategy tailored for heterogeneous devices and asynchronous communications that is adept at detecting and remediating potential attacks. The primary contributions are as follows.

Pioneering Asynchronous and Heterogeneous Federated Learning with Anomaly Resilience. To the best of our knowledge, we are the first to propose a reliable federated learning algorithm that supports asynchronous communication and heterogeneous models, also facilitating the detection and recovery of anomalous clients in practical FL scenarios.

Adaptive Sparse Recovery and Anomaly Mitigation for Robust Federated Learning. Our preliminary experiments showed client-induced errors in asynchronous federated learning, lowering accuracy without efficient model recovery. We developed a new method: the server caches past updates to fill missing parameters in sparse updates, assigning weights based on staleness. Consistent historical updates address Non-IID data. Detecting discrepancies between local and global updates helps identify attackers, and the model is reconstructed using the Cauchy mean value theorem.

Convergence Proven, Suspicion Quantified for Secure Federated Learning. We demonstrate an upper bound of convergence for the FedHAN algorithm and identify a suspicion score threshold to distinguish benign from malicious clients. (1) Model staleness and size significantly affect convergence speed; (2) FedHAN's suspicion scores effectively differentiate between attackers and benign clients.

Elevating Accuracy and Resilience Against Federated Adversaries. We conducted extensive tests and found that FedHAN increased model accuracy by 7.86% over recent federated learning algorithms. In three poisoning attack scenarios, including backdoor attacks, its detection and recovery outperformed leading methods by 12% and 7.16%, demonstrating the robustness of FedHAN in complex real-world settings.

2 Background and Related Work

2.1 Poisoning Attacks in SAFL

Federated learning is vulnerable to attacks that skew local data or update models, such as backdoor [Bagdasaryan *et al.*, 2020] and trim [Fang *et al.*, 2020] attacks, affecting the accuracy and reliability of the model. Detecting these threats involves analyzing differences in the feature distribution. Techniques include FABA [Xia *et al.*, 2019], which flags the most deviant updates; FoolsGold [Fung *et al.*, 2020], which assesses client similarity; and FLDetector [Zhang *et al.*, 2022], which applies the Cauchy mean value theorem. However, their effectiveness wanes in real-world settings. FedHAN bolsters FL systems by integrating caching and update imputation to enhance robustness and threat detection in complex environments.

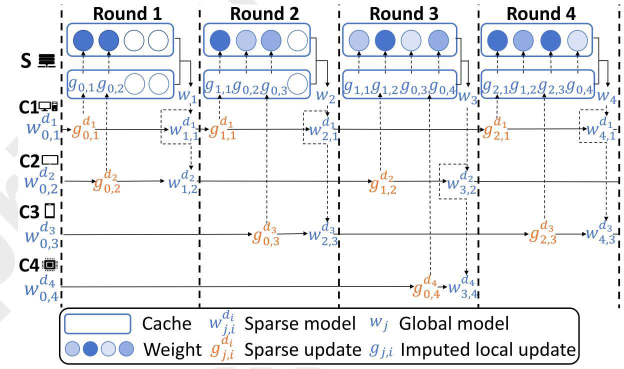


Figure 1: Cache-based FedHAN model update mechanism architecture.

2.2 Machine Unlearning

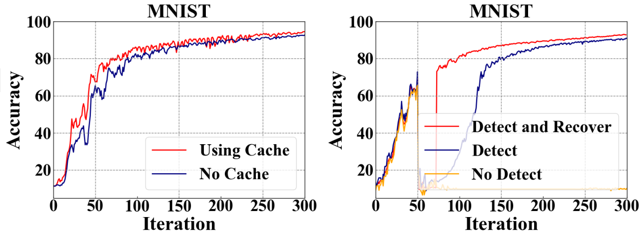
Detection methods often take multiple rounds, allowing the global model to already be compromised. A key challenge is countering malicious clients post-detection. Retraining with only benign clients is an option, but it is costly in terms of communication. Current recovery strategies cut communication costs by leveraging historical data. For example, FedRecovery [Cao *et al.*, 2023] uses the Cauchy mean value theorem to predict updates, FedEraser [Liu *et al.*, 2021] adjusts updates with past data, and Crab [Jiang *et al.*, 2025] picks crucial historical info using KL divergence and cosine similarity. These strategies presume detection of malicious clients and do not integrate well with mainstream detection, thus reducing real-world efficacy.

FedHAN addresses these challenges with a pioneering algorithm that merges detection and recovery of Byzantine attacks in FL, boosting system robustness and utility in real-world federated settings.

3 Methodology

3.1 Problem Framework

We explore a federated learning framework that protects a central server and N clients (see Figure 1). Initially, each client submits their model width $d_{0,i}$, and the server generates a tailored initial model $w_{0,i}^{d_{0,i}}$ for each client. These models are sent back to the clients, who then use stochastic update descent (SGD) with their local data for updates. Upon completing their updates, clients upload them directly to the



(a) (b)

Figure 2: Motivation experiments. (a) Effect of historical update integration on model training convergence. (b) Impact of robust recovery algorithms on adversarial robustness after attack detection.

server. The server keeps a local update cache of size N, storing the latest imputed updates from each client. An aggregation round is initiated after the cache is updated K times, leading to an update of the global model:

$$w_{j+1} = w_j - \eta_j \sum_{i=1}^{N} p_{j,i}\bar{g}(w_{j,i}, \xi_{j,i}), \quad (1)$$

where η_j is the learning rate for the j-th iteration, $\bar{g}(w_{j,i})$ signifies the latest local update for client i at round j, and $\xi_{j,i}$ indicates the data sample employed by the client in this round. The weight $p_{j,i}$ indicates the significance of the update of the client i in the aggregation for round j, influenced by the Non-IID data levels, the staleness of the update and the status of the cache update.

3.2 Definitions

Definition 1. *[Hong* et al.*, 2022] To describe the size of the model that each client can support, the model size for each client is defined as the ratio of the hidden channel width in its local model to the width of the full-scale global model, denoted as $d_{t,i}$, where $d_{t,i} \in (0, 1]$. Specifically, when $d_{t,i} = 1$, the size of $w^{d_{t,i}}$ is equivalent to the size of the global model.*

Definition 2. *[Wang* et al.*, 2023] The operation $\odot$ is defined as the element-wise product, representing the multiplication of corresponding elements in two tensors. Let w denote the global model, $w^{d_{t,i}}$ denote the customized sparse model for client i in round t, $d_{t,i}$ represent the width of the client model i in round t, and $m^{d_{t,i}}$ denote the shape of the model corresponding to $d_{t,i}$. According to Definition 2, the sparse model for a client can be calculated as $w^{d_{t,i}} = w \odot m^{d_{t,i}}$.*

3.3 Motivation

In asynchronous FL, each training round involves only K clients, causing a "partial client participation bias" problem. This intensifies model drift due to data heterogeneity, as the global model tends to overfit the data of the participating clients, reducing its generalization capability. To tackle this, we propose selecting some historical updates based on cosine similarity and incorporating them into the training process. Preliminary experiment results (Figure 2a) show that the red curve, which includes historical updates, exhibits reduced fluctuations and faster convergence compared to the

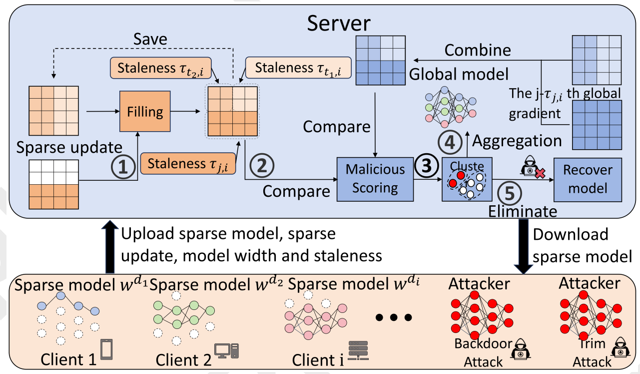


Figure 3: Update imputation, attack detection, and recovery architecture in FedHAN.

blue curve with stale updates. This suggests that the approach can enhance model performance and training efficiency.

Traditional defenses against poisoning attacks generally entail eliminating the identified malicious clients and proceeding with training using the remaining clients. However, this approach does not fully negate their adverse effects. To tackle this problem, we propose adding a recovery process immediately after detecting a malicious attack, aiming to reduce its effects. Preliminary experimental results (Figure 2b) confirmed that models using both detection and recovery methods outperform those relying only on detection strategies.

3.4 Overall Design of FedHAN

The FedHAN system is composed of three main modules, including caching and model aggregation, attack detection, and model recovery (see Figure 3). This modular structure allows the system to handle challenges efficiently and safely. Algorithm 1 (see Appendix **A**) details the FedHAN execution through these steps. (1) Sparse Model Customization. The server generates a client-specific mask $m^{d_{t,i}}$ based on the capacity of the client, customizes a sparse model $w^{d_{t,i}} = w \odot m^{d_{t,i}}$, and sends it to the client. (2) Local Training and Upload. Clients train their sparse models on local data and upload the sparse update $g(w_{t,i}^{d_{t,i}}, \xi_{t,i})$ to the server. (3) Asynchronous Reception and Update Imputation. The server collects sparse updates from the latest K clients, pads missing updates using past data to get $\bar{g}(w_{j,i})$ (see Equation (2)), forms a weight matrix based on update timing and client masks (see Equation (3)) and updates caches. (4) Malicious Update Detection. The server identifies malicious clients by checking the Euclidean distance $s_{j,i} = \|\check{g}(w_{j,i}) - \bar{g}(w_{j,i})\|_2$ and classifies updates with DBSCAN and k-means. Malicious clients are removed if detected; otherwise, this step is skipped. (5) Global Model Recovery. Using the Cauchy median theorem, the server restores a global model, excluding malicious effects with the help of benign clients and historical data. (6) Sparse Model Imputation and Aggregate. Using the latest imputed local update to impute a spare model update, followed by aggregation to form $\bar{g}(w_j)$. (7) Historical Updates Select and Aggregate. The server selects some unreceived updates from the cache based on cosine similarity, followed by aggregation to form $\tilde{g}(w_j)$. (8) Global Model Update. Finally, the server computes and applies the global update using Equation (7).

3.5 Implementation Details of FedHAN

Sparse Model Imputation and Aggregate. To mitigate the impact of Non-IId data and accelerate model convergence, we impute the sparse model updates uploaded by clients. We define the update rule for the latest update $\bar{g}(w_{j,i})$ of the client i. If the update of the current client i, $g(w_{t,i}^{d_{t,i}}, \xi_{t,i})$, reaches the server, i.e., $i \in C_j$, it will be incorporated into the latest imputed local updates, specifically $g(w_{t,i}^{d_{t,i}}, \xi_{t,i}) \cup \bar{g}(w_{j-1,i}) \odot \tilde{m}^{d_{t,i}}$; otherwise, the latest imputed update will be the imputed update from the most recent round.

$$\bar{g}(w_{j,i}) = \begin{cases} g(w_{t,i}^{d_{t,i}}, \xi_{t,i}) \cup (\bar{g}(w_{j-1,i}) \odot \tilde{m}^{d_{t,i}}) & \text{if } i \in C_j, \\ \bar{g}(w_{j-1,i}) & \text{if } i \in C_j^-, \end{cases} \tag{2}$$

where $g(w_{t,i}^{d_{t,i}}, \xi_{t,i})$ represents the local sparse update generated by client i using the global model from round t, and $\bar{g}(w_{j-1,i})$ denotes the imputed update from the previous round.

The weight matrix $M_{j,i}$ represents the weights of different parts of $\bar{g}(w_{j,i})$. When the latest imputed update is updated, the newly imputed part is assigned a weight of $\beta^{\tau_{j,i}}$ based on its staleness, while the other parts decay exponentially with a factor of β. If no update occurs, the weight matrix $M_{j,i}$ decays exponentially with the factor β.

$$M_{j,i} = \begin{cases} \beta^{\tau_{j,i}} m^{d_{t,i}} \cup (\beta M_{j-1,i} \odot \tilde{m}^{d_{t,i}}) & \text{if } i \in C_j, \\ \beta M_{j-1,i} & \text{if } i \in C_j^-, \end{cases} \tag{3}$$

where $\beta^{\tau_{j,i}} m^{d_{t,i}}$ represents the weight of $g(w_{t,i}^{d_{t,i}}, \xi_{t,i})$, while $\beta M_{j-1,i} \odot \tilde{m}^{d_{t,i}}$ denotes the weight of the remaining parts.

We assign a $M_{j,i}$ weight to the sparse model update $g(w_{t,i}^{d_{t,i}}, \xi_{t,i})$ uploaded by the client, since it participates directly in the model update, while the remaining unupdated part is assigned a separate $\alpha M_{j,i}$ weight. The calculation is as follows.

$$p'_{j,i} = M_{j,i}(m^{d_{t,i}} \cup \alpha \tilde{m}^{d_{t,i}}), \tag{4}$$

where $p'_{j,i}$ represents the final weight of the imputed update.

Finally, we perform a weighted aggregation of $\bar{g}(w_{j,i})$ to form $\bar{g}(w_j)$.

$$\bar{g}(w_j) = \sum_{i=1}^{K} p_{j,i} \bar{g}(w_{j,i}), \tag{5}$$

where $p_{j,i} = p'_{j,i} / \sum_{i=1}^{K} p'_{j,i}$.

Historical Updates Select and Aggregate. To reduce the impact of Non-IID data on model training, more local updates are needed for aggregation. Updates from non-participating clients with lower staleness and consistent update directions can be selected to accelerate training.

The server first calculates the estimated unbiased update $\bar{g}(w_j)$ aggregated from the local updates of the first K clients and computes the cosine similarity $sim_{j,i}$ between it and the updates $\bar{g}(w_{j,i})$ from the remaining clients in the cache. If $sim_{j,i} \geq sim_{min}$, the update is included in the aggregation, where sim_{min} is a predefined threshold. Updates with similarity lower than sim_{min} are discarded as inconsistent. The formula is as follows.

$$sim_{j,i} = \cos(\bar{g}(w_{j,i}), \bar{g}(w_j)), \tag{6}$$

where $\bar{g}(w_j)$ represents the estimated unbiased update, and $\bar{g}(w_{j,i})$ represents the historical update of the remaining clients.

Subsequently, to further reduce the impact of Non-IID data on model training, we incorporate the global update $\tilde{g}(w_j)$, aggregated from the selected client updates, into the global update $\bar{g}(w_j)$ derived from the K most recent updates, scaled by a factor α. The formula is as follows.

$$g(w_j) = \bar{g}(w_j) + \alpha \tilde{g}(w_j), \tag{7}$$

where $\alpha > 0$ is a constant, while $\bar{g}(w_j)$ represents the aggregated result of the K most recent updates received by the server during the j-th iteration, and $\tilde{g}(w_j)$ denotes the global update from the selected client updates.

Malicious Client Detection. To detect malicious clients, the server maintains a latest imputed global update for each client. The suspicious score for each client is evaluated as $s_{j,i} = \|\breve{g}(w_{j,i}) - \bar{g}(w_{j,i})\|_2$, and these scores are collected into the set S_j. After collecting the scores, we apply the density-based clustering algorithm DBSCAN. If S_j can be divided into multiple clusters or contains noise points according to the DBSCAN algorithm, we further apply the k-means algorithm to partition the suspicious scores S_j into two clusters. Finally, clients belonging to the cluster with the highest average suspicious score are identified as malicious clients and removed from the current round.

We determine the global model $g(w_{j-\tau_{j,i}})$ in round $t = j - \tau_{j,i}$ based on the staleness $\tau_{j,i}$ of the sparse update uploaded by the client i. The latest imputed global update $\breve{g}(w_{j,i})$ is updated synchronously with the latest imputed sparse update $\bar{g}(w_{j,i})$. Once the latest imputed sparse update $\bar{g}(w_{j,i})$ is updated, the latest imputed global update $\breve{g}(w_{j,i})$ is updated according to the following rule as follows.

$$\breve{g}(w_{j,i}) = \begin{cases} g(w_{j-\tau_{j,i}}) \odot m^{d_{t,i}} \cup \breve{g}(w_{j-1,i}) \odot \tilde{m}^{d_{t,i}}, & \text{if } i \in C_j, \\ \breve{g}(w_{j-1,i}), & \text{else}, \end{cases} \tag{8}$$

where $t = j - \tau_{j,i}$, and $g(w_{j-\tau_{j,i}})$ represents the global model from round t, and $\breve{g}(w_{j,i})$ denotes the latest imputed global update for client i in round j.

Heterogeneous Model Recovery. In the T-th round of detection, the algorithm identified and removed all malicious clients, then rolled back m rounds to select a clean model w_{T-m} as the starting point for a new training cycle, ensuring that subsequent training occurs in a reliable environment. To maintain training continuity and efficiency, the recovery algorithm combined two strategies: precise updates with benign clients following the FedHAN protocol and estimated

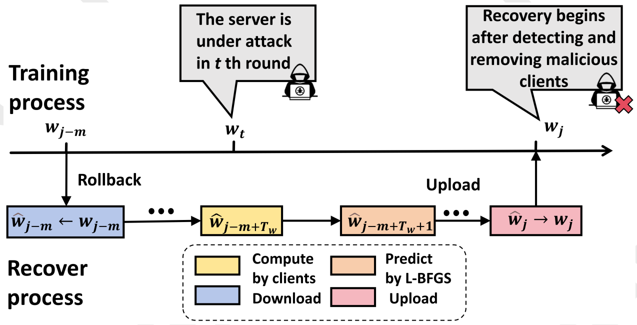


Figure 4: Model recovery process details.

updates using historical data from rounds $T-m$ to T, reducing the reliance on real-time participation. To prevent error accumulation, clients performed precise calculations during the early recovery phase and at fixed intervals, promptly correcting errors and enhancing recovery quality (see Figure 4). This innovative design balanced estimated updates and precise computations, minimizing error risks and ensuring stable and reliable global model recovery.

We utilize the L-BFGS algorithm [Liu and Nocedal, 1989] to approximate the Hessian matrix in the heterogeneous model recovery. Specifically, each iteration draws on changes in the global model and updates from previous iterations for this purpose. The global model deviation in iteration t is $\Delta w_t = \hat{w}_t - w_t$, indicating the difference between the recovered and original global models. For the i-th client, the model update deviation is $\Delta g_t^i = \bar{g}(\hat{w}_{t,i}, \xi_{t,i}) - \bar{g}(w_{t,i}, \xi_{t,i})$, showing the variance from the exact update. The algorithm keeps a buffer of global model deviations, $\Delta W_t = [\Delta w_{b_1}, \Delta w_{b_2}, \ldots, \Delta w_{b_s}]$, and a buffer of client update deviations, $\Delta G_t^i = [\Delta g_{b_1}^i, \Delta g_{b_2}^i, \ldots, \Delta g_{b_s}^i]$, where s is the buffer size. Based on the integral version of the Cauchy mean value theorem, we can calculate the estimated model updates g_j^i as follows.

$$\hat{g}(\hat{w}_{j,i}, \xi_{j,i}) = \bar{g}(w_{j,i}, \xi_{j,i}) + \hat{H}_{j,i}(\hat{w}_j - w_j), \tag{9}$$

where $\hat{H}_{j,i}(\hat{w}_j - w_j)$ is calculated by the L-BFGS algorithm, and $\bar{g}(w_{j,i}, \xi_{j,i})$ represents the imputed client update during the training process.

4 Theoretical Analysis

4.1 Complexity Analysis

To mitigate the error caused by partial client participation, the server needs to store the latest updates, weights, masks, and global updates for all clients. The storage cost for this part is $O(4Np)$, where p represents the number of parameters in the global model. Furthermore, the recovery algorithm requires historical updates from the most recent m rounds to accelerate the recovery process, which incurs a storage cost of $O(mKp)$, where K denotes the number of clients participating in asynchronous aggregation per round. Thus, the overall space complexity of FedHAN is $O(mKp+4Np)$.During the recovery phase, FedHAN selects historical updates based on cosine similarity. Assuming that the number of clients that satisfy $\tau_{j,i} \le \tau_{max}$ is s, the time complexity of this operation is $O(spT)$. In addition, some communication costs are introduced for the clients. These communication costs mainly depend on the number of rounds in which clients are required to compute model updates. The additional communication cost per client can be expressed as $O(T_w + \frac{m-T_w}{T_c})$, where T_w represents the number of rounds that require precise updates in the early stage of the recovery process, $\frac{m-T_w}{T_c}$ represents the number of rounds that require periodic precise updates during recovery. Thus, the additional communication cost per client is $O(T_w + \frac{m-T_w}{T_c})$.

4.2 Convergence Analysis

Theorem 1. *Assume Assumptions 1, 2, 3, 4 hold. Learning rate satisfies that $\eta \le \frac{1}{Ls_j}$ and subjects to $\mathcal{M}_j \ge 0$, where*

$$\mathcal{M}_j = (\frac{\eta_j s_j}{2} - \sum_{l=j}^{J} \eta_l \sum_{t=j}^{l-1} \eta_t^2 \sum_{k=1}^{t} \alpha^{t-j} I_{l,k,t}).$$

Then, we can obtain the following convergence result

$$\frac{1}{J}\sum_{j=1}^{J} \mathcal{M}_j ||\nabla F(w_j)||_2^2 \le \frac{1}{J}\sum_{j=1}^{J}(\frac{3\eta_j s_j}{2}C + \eta_j \sigma_c^2 \sum_{l=1}^{j} \sum_{t=l}^{j-1} \eta_t^2 s_t I_{l,k,t}) + \frac{F(w_1) - F(w^*)}{J},$$

where $C = G^2 + \sigma_e^2 \sum_{i=1}^{K} \mathbb{E}[p_{l,i}]$ *and* $I_{l,k,t} = 3L^2B^2 s_t \alpha^{l-k}(l-\tau_k)$.

4.3 Theoretical Analysis on Suspicious Scores

Theorem 2. *Suppose that the update of each client's loss function is L-smooth, FedHAN is used as the aggregation rule, the learning rate α satisfies $\alpha < \frac{1}{(N+2)L}$ (N is the window size). Suppose that malicious clients perform an untargeted model poisoning attack in each iteration by reversing the true model updates as the poisoning ones, that is, each malicious client i sends $-g(w_{j,i})$ to the server in each iteration t. Then we find that the expected suspicious score of a benign client is smaller than that of a malicious client in each iteration t. Formally, we have the following inequality.*

$$E(s_i^t) < E(s_a^t), \forall i, \tag{10}$$

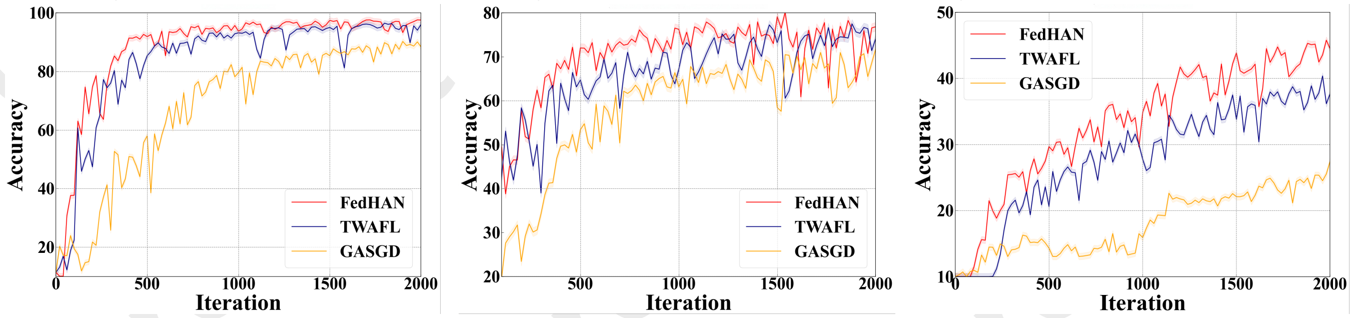
where the expectation E is taken with respect to the randomness in the clients' local training data, $\mathcal{B}$ is the set of benign clients, and M is the set of malicious clients.

The theoretical proofs of Theorem 1 and Theorem 2 are provided in Appendix **B**.

5 Experiments and Discussion

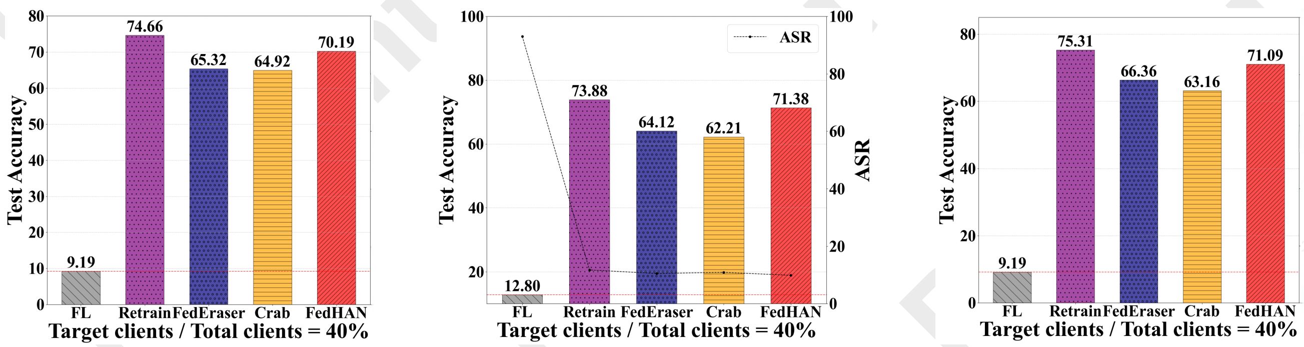
5.1 Experimental Setup

FedHAN is evaluated using MNIST, FMNIST, and CIFAR-10 datasets with trim attack, backdoor attack and label-flipping (LF) attack. We compare asynchronous federated learning methods like TWAFL, SASGD, Byzantine defense methods such as FoolsGold, FLDetector, and recovery strategies like Retrain, FedEraser, Crab. Experiments involve 2000 communication rounds with local training epochs and a batch size of 4, using SGD at a 0.005 learning rate. The model has two convolutional and two fully connected layers. Data heterogeneity is simulated with Dirichlet distribution values of 0.3 and 0.8; model heterogeneity uses 0.2 and 0.5 mask levels. The momentum is 0.2, decay rate is 0.9. Further details are in Appendix **C**.



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10

Figure 5: The performance comparison of three different algorithms under $N/K = 1000/20$ on the MNIST, Fashion-MNIST and CIFAR-10 dataset.



(a) Trim attack (b) Backdoor attack (c) LF attack

Figure 6: The test accuracy (presented in the bar chart) and ASR (presented in the line chart) on MNIST dataset after recovering from trim attack, backdoor attack and LF attack respectively when the attack intensity is at 40%.

5.2 Comparison of FedHAR with Other Models

Results and Analysis. Figure 5 shows FedHAN outperforming TWAFL and GASGD on MNIST, Fashion-MNIST, and CIFAR-10 datasets, with quicker accuracy gains and smoother training curves, highlighting its stability. FedHAN remains robust against staleness ($N/K = 1000/20$), thanks to its client selection mechanism minimizing errors from non-participating clients. While TWAFL balances update information well, Table 1 corroborates FedHAN's superiority in accuracy, speed, and robustness, effectively tackling heterogeneity and Non-IID issues. Its design enhances model generalization and accuracy across data distributions, proving its value for real-world federated learning.

5.3 Ablation Study of the Detection Algorithm

Detection Results. Table 2 shows that FedHAN excels over other defenses in the MNIST, Fashion-MNIST and CIFAR-10 datasets. It achieves 100% effectiveness against Trim attacks, has impressive detection rates of 91% for backdoor attacks, and 90% for label flipping attacks on MNIST, with low false negatives (15%, 12%) and false positives (5%, 7%). Although slightly less effective in CIFAR-10, FedHAN still proves to be a solid defense.

Dataset	Mask level	Dirichlet distribution	N/K=1000/20		
			FedHAN	TWAFL	SASGD
MNIST	0.5	0.8	98.91	95.37	90.14
		0.3	97.13	93.47	90.25
	0.2	0.8	97.77	94.87	89.54
		0.3	95.19	91.05	84.25
Fashion MNIST	0.5	0.8	82.93	78.45	65.75
		0.3	80.41	73.31	52.05
	0.2	0.8	77.44	74.67	63.74
		0.3	74.62	70.37	58.63
CIFAR-10	0.5	0.8	47.31	41.42	31.51
		0.3	45.16	37.89	25.05
	0.2	0.8	45.87	37.98	27.76
		0.3	43.94	34.49	21.94

Table 1: Prediction accuracy of FedHAN at the level of staleness ($N/K = 1000/20$), model heterogeneity, and Non-IID degree.

5.4 Ablation Study of the Recovery Algorithm

Test Data Accuracy. Figure 6 demonstrates that in analyzing the MNIST dataset at the 60th recovery round, increasing

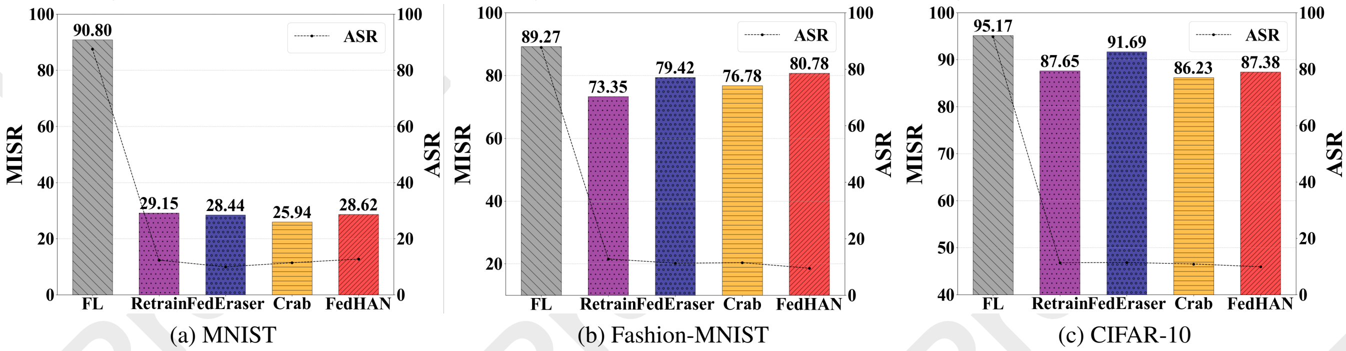


Figure 7: The MISR (presented in the bar chart) and ASR (presented in the line chart) on MNIST, Fashion-MNIST and CIFAR-10 after recovering from backdoor attack.

Dataset	Detector	Trim attack			Backdoor attack			LF attack		
		DACC	FPR	FNR	DACC	FPR	FNR	DACC	FPR	FNR
MNIST	FoolsGold	0.92	0.09	0.03	0.83	0.31	0.44	0.79	0.43	0.36
	FLDetector	0.96	0.04	0.02	0.84	0.15	0.11	0.81	0.32	0.11
	FedHAN	**1.00**	**0.00**	**0.00**	**0.91**	**0.15**	**0.05**	**0.90**	**0.12**	**0.07**
Fashion-MNIST	FoolsGold	0.91	0.08	0.04	0.77	0.67	0.15	0.72	0.64	0.22
	FLDetector	0.95	0.05	0.02	0.80	0.41	0.10	0.81	0.30	0.11
	FedHAN	**1.00**	**0.00**	**0.00**	**0.90**	**0.14**	**0.05**	**0.87**	**0.21**	**0.13**
CIFAR-10	FoolsGold	0.93	0.09	0.02	0.63	0.83	0.32	0.65	0.81	0.33
	FLDetector	0.95	0.05	0.02	0.75	0.78	0.17	0.75	0.71	0.15
	FedHAN	**1.00**	**0.00**	**0.00**	**0.87**	**0.12**	**0.08**	**0.82**	**0.31**	**0.14**

Table 2: Detection accuracy, FPR, FNR comparisons of FedHAN versus baseline approaches against trim attack, backdoor attack and label flipping attack.

malicious client percentages caused a significant drop in test accuracy (e.g., 12.8% with 40% malicious clients in backdoor attacks). However, the FedHAN algorithm consistently outperformed other recovery methods, demonstrating its strength in tackling attacks and enhancing model resilience.

Attack Success Rate (ASR). Figure 7 illustrates that the Attack Success Rate of the Recovery Algorithm (ASR) dropped to approximately 12.7%, indicating a strong defense against backdoor threats. Also, Membership Inference Attacks (MIA) confirmed its prowess in mitigating privacy threats, with the Membership Inference Success Rate (MISR) reduced to 28.62% on MNIST dataset, underscoring the systeMs capability in preserving data privacy and model security.

5.5 Discussion

FedHAN effectively addresses outdated updates and Non-IID data in global models through update imputation, adaptive weighting, and consistent historical update selection. It uses past data to supplement updates and adjusts the influence of each update based on its staleness, allowing for meaningful contributions to the global model.

By aggregating updates using cosine similarity, it ensures diverse client input, boosting model accuracy by 9.56% (see Table 1). In federated settings with model heterogeneity and asynchronous communication, FedHAN efficiently detects malicious clients by imputing updates and analyzing deviations between local and global updates, using clustering algorithms like DBSCAN and k-means. It enhances detection accuracy by 12% compared to traditional methods (see Table 2). With historical update imputation, FedHAN reconstructs a robust global model post-malicious client detection, outperforming other methods by 7.26% without significant accuracy loss (see Figure 6).

6 Conclusion

Under realistic circumstances, federated learning faces significant obstacles, such as Non-IID data, asynchronicity, model heterogeneity, and poisoning attacks. We propose FedHAN, a cache-based semi-asynchronous algorithm designed to enhance robustness. FedHAN employs sparse model filling, adaptive weight distribution, and selective historical updates to handle stale updates and Non-IID data effectively. This method keeps a padded global update while incorporating client-sparse updates, aiding in the accurate identification of malicious attacks. By applying the Cauchy mean value theorem, FedHAN neutralizes the effects of such attacks, ensuring model reliability. We demonstrate FedHAN's convergence and its proficiency in detecting attacks, with experiments across three datasets confirming its superior accuracy and robustness, underscoring its practical utility.

A Algorithm

Algorithm 1 FedHAN

Require: learning rate η_0, number of updates received by server K, weighted parameter β, momentum α
Ensure: Optimal solution w^*

<u>**Server side:**</u>
Initialize model parameter w_0 and iteration $j = 1$
Clients send model width $d_{t,i}$ to server.
Server customizes $w_j^{d_{t,i}} = w_j \odot m^{d_{t,i}}$ to all clients.
while the stopping criteria is not satisfied **do**
  $stage = 1$
  $\bar{g}(w_j) \leftarrow SMIAA(\cdot)$
  $\tilde{g}(w_j) \leftarrow HUSAA(\cdot)$
  $g(w_j) = \bar{g}(w_{j,i}) + \alpha\tilde{g}(w_{j,i})$
  Call Detect($\cdot$) to detect malicious clients.
  if Detected malicious clients **then**
    $stage = 2$
    $\tilde{w_j}$= Recover($\cdot$)
    $w_j \leftarrow \tilde{w_j}$
    continue
  end if
  $w_{j+1} \leftarrow w_j - n_j g(w_j)$
  $j \leftarrow j + 1$
end while

<u>**Client side:**</u>
Receive model $w_t^{d_{t,i}}$ from server at round t.
Perform update descent to get $g(w_{t,i}^{d_{t,i}}, \xi_{t,i})$ based on samples $\xi_{t,i}$.
Upload $g(w_{t,i}^{d_{t,i}}, \xi_{t,i})$ and model widthd_i.

B Proofs of Theorem 1 and Theorem 2

B.1 Assumption

Assumption 1. *Lipschitz Continuity. Objective function $F(\cdot)$ satisfies L-Lipschitz continuity,$\forall w_1, w_2, \exists$constant L*

$$\|F(w_1) - F(w_2)\| \leq \nabla F(w)^T \cdot \frac{L}{2}\|w_1 - w_2\|_2^2. \tag{11}$$

Assumption 2. *(Client-Level Unbiased Update).The update $g(w_j, \xi_{j,i})$ of client i is a client-level unbiased update which means that the expectation of update $g(w_j, \xi_{j,i})$ is equal to $\nabla F_i(w_j)$ as follows.*

$$E[g(w_j, \xi_{j,i})] = \nabla F_i(w_j). \tag{12}$$

Assumption 3. *(Updates With Bounded Variance).The update $g(w_j, \xi_{j,i})$ of client i has client-level bounded variance: $\exists$ constants σ_c, M_c,*

$$E\left[\|g(w_j, \xi_{j,i}) - \nabla F_i(w_j)\|_2^2\right] \leq \frac{\sigma_c^2}{m} + \frac{M_c}{m}\|\nabla F_i(w_j)\|_2^2, \tag{13}$$

where $\nabla F_i(w_j)$ is the unbiased update of client i. To guarantee the convergence of the model, we also need to assume $\nabla F_i(w_j)$ satisfies global-level bounded variance: $\exists$ constant G,

$$\|\nabla F(w_j) - \nabla F_i(w_j)\|_2^2 \leq G^2. \tag{14}$$

We have explained the rationale behind using historical update imputation and clip techniques to enhance prediction accuracy and stabilize the training process in previous sections. In this section, we further analyze the convergence rate of the proposed algorithm in which the loss function is non-convex, considering both staleness and heterogeneity. We begin with a proof sketch for the proposed FedHAN algorithm. Firstly, we connect the local update $\bar{g}(w_{l,i}, \xi_{l,i})$ with update $g(w_l)$ by clipping $\bar{g}(w_{l,i}, \xi_{l,i})$, we have

$$\|\bar{g}(w_j, \xi_{j,i})\|_2^2 \leq B^2\|g(w_j)\|_2^2. \tag{15}$$

Secondly, after alleviating the effect of Non-IID data and staleness, for the reason that the estimated update $\bar{g}(w_l)$ is close to the unbiased updates, we can assume

$$\|\bar{g}(w_l) - \nabla F(w_l)\|_2^2 \leq \sigma_c^2, \quad \exists\sigma_c^2, \tag{16}$$

which bridges the connection between $\bar{g}(w_l)$ and $\nabla F(w_l)$.

B.2 Proof of Theorem 1

Proof of Theorem 1. Based on Assumption 1 and update rule $w_{j+1} = w_j - \eta_j \sum_{l=1}^{j} \alpha^{j-l} \sum_{i=1}^{K} p_{l,i}\bar{g}(w_{l,i}, \xi_{l,i})$, we have

$$\begin{aligned}
&F(w_{j+1}) - F(w_j) \\
&\leq \nabla F(w_j)(w_{j+1} - w_j) + \frac{L}{2}\|w_{j+1} - w_j\|_2^2 \\
&= -\frac{\eta_j}{2}\sum_{l=1}^{j}\alpha^{j-l}\sum_{i=1}^{K}p_{l,i}(||\nabla F(w_j)||_2^2 \\
&\quad + ||\bar{g}(w_{l,i}, \xi_{l,i})||_2^2 - ||\nabla F(w_j) - \bar{g}(w_{l,i}, \xi_{l,i})||_2^2) \\
&\quad + \frac{L}{2}\eta_j^2||\sum_{l=1}^{j}\alpha^{j-l}\sum_{i=1}^{K}p_{l,i}\bar{g}(w_{l,i}, \xi_{l,i})||_2^2.
\end{aligned}$$

Define $s_j = \sum_{l=1}^{j}\alpha^{j-l} = \frac{1-\alpha^j}{1-\alpha}$ and take expectation on both sides of the above equation:

$$\begin{aligned}
&\mathbb{E}[F(w_{j+1})] - F(w_j) \\
&\leq -\frac{\eta_j}{2}\sum_{l=1}^{j}\alpha^{j-l}\mathbb{E}[\sum_{i=1}^{K}p_{l,i}(||\nabla F(w_j)||_2^2 + ||\bar{g}(w_{l,i}, \xi_{l,i})||_2^2 \\
&\quad - \|\nabla F(w_j) - \bar{g}(w_{l,i}, \xi_{l,i})\|_2^2)] \\
&\quad + \frac{L}{2}\eta_j^2 s_j\sum_{l=1}^{j}\alpha^{j-l}\mathbb{E}[||\sum_{i=1}^{K}p_{l,i}\bar{g}(w_{l,i}, \xi_{l,i})||_2^2] \qquad (17) \\
&\leq -\frac{\eta_j}{2}\sum_{l=1}^{j}\alpha^{j-l}\mathbb{E}[\sum_{i=1}^{K}p_{l,i}(\|\nabla F(w_j)\|_2^2 + ||\bar{g}(w_{l,i}, \xi_{l,i})||_2^2 \\
&\quad - ||\nabla F(w_j) - \bar{g}(w_{l,i}, \xi_{l,i})||_2^2)] \\
&\quad + \frac{L}{2}\eta_j^2 s_j\sum_{l=1}^{j}\alpha^{j-l}\mathbb{E}[\sum_{i=1}^{K}p_{l,i}||\bar{g}(w_{l,i}, \xi_{l,i})||_2^2] \qquad (18) \\
&\leq -\frac{\eta_j s_j}{2}||\nabla F(w_j)||_2^2 + \frac{\eta_j}{2}\sum_{l=1}^{j}\alpha^{j-l} \\
&\quad \underbrace{\mathbb{E}[\sum_{i=1}^{K}p_{l,i}||\nabla F(w_j) - \bar{g}(w_{l,i}, \xi_{l,i})||_2^2]}_{\mathcal{A}}. \qquad (19)
\end{aligned}$$

Equations (17) and (18) are derived on the basis of Jensen's inequality. Equation (19) is derived because the learning rate satisfies $\eta_j \leq \frac{1}{Ls_j}$. With respect to term $\mathcal{A}$,

$$\begin{aligned}\mathcal{A} =& \mathbb{E}[\sum_{i=1}^{K} p_{l,i} \|\nabla F(w_j) - \nabla F_i(w_j) + \nabla F_i(w_j) \\ &- \nabla F_i(w_{\tau(l)}) + \nabla F_i(w_{\tau(l)}) - \bar{g}(w_{l,i}, \xi_{l,i})\|_2^2] \\ &\leq 3G^2 + 3L^2 \underbrace{\mathbb{E}[\sum_{i=1}^{K} p_{l,i} \|w_j - w_{l-\tau_{l,i}}\|_2^2]}_{\mathcal{B}} + 3\sigma_e^2 \sum_{i=1}^{K} \mathbb{E}[p_{l,i}],\end{aligned} \tag{20}$$

Equation (20) is derived based on Assumptions 3.1 and 3.3. Define $\tau_l = \max(1, l - \tau_{\max})$. With respect to term $\mathcal{B}$,

$$\begin{aligned}\mathcal{B} \leq& E[\sum_{i=1}^{K} p_{l,i} \|w_j - w_{l-\tau_{l,i}}\|_2^2] \\ \leq& E[\sum_{i=1}^{K} p_{l,i} (j - \tau_l) \sum_{t=\tau_l}^{j-1} \eta_t^2 \| \sum_{k=1}^{t} \alpha^{t-k} \sum_{q=1}^{K} p_{k,q} \bar{g}(w_{k,q}, \xi_{k,q})\|_2^2] \\ \leq& (j - \tau_l) E[\sum_{t=\tau_l}^{j-1} \eta_t^2 s_t \sum_{k=1}^{t} \alpha^{t-k} \sum_{i=1}^{K} p_{k,i} \|\bar{g}(w_{k,i}, \xi_{k,i})\|_2^2] \\ \leq& 2(j - \tau_l) E[\sum_{t=\tau_l}^{j-1} \eta_t^2 s_t \sum_{k=1}^{t} \alpha^{t-k} B^2 \|\nabla F(w_k)\|_2^2 \\ &+ 2\sigma_c^2 B^2 \sum_{t=l}^{j-1} (j - \tau_l) \eta_t^2 s_t^2.\end{aligned} \tag{21}$$

By replacing Equations (20), (21) into Equation (19), we have

$$\begin{aligned}&\mathbb{E}[F(w_{j+1})] - F(w_j) \\ &\leq -\frac{\eta_j s_j}{2} ||\nabla F(w_j)||_2^2 + \frac{3}{2} \eta_j s_j G^2 + \frac{3}{2} \eta_j s_j \sigma_e^2 \sum_{i=1}^{K} \mathbb{E}[p_{l,i}]; \\ &+ 3\eta_j L^2 B^2 \sum_{l=1}^{j} \sum_{t=l}^{j-1} \sum_{k=1}^{t} \alpha^{j-k} (j - \tau_k) \eta_l^2 s_t \alpha^{t-l} \|\nabla F(w_l)\|_2^2 \\ &+ 3\eta_j L^2 \sigma_c^2 B^2 \sum_{l=1}^{j} \alpha^{j-l} \sum_{t=l}^{j-1} (j - \tau_l) \eta_t^2 s_t^2.\end{aligned}$$

Taking summation with respect to j on both sides, we obtain

$$\begin{aligned}&F(w^*) - F(w_1) \\ &\leq -\sum_{j=1}^{J} \frac{\eta_j s_j}{2} ||\nabla F(w_j)||_2^2 + \frac{3}{2} \sum_{j=1}^{J} \eta_j s_j (G^2 + \sigma_e^2 \sum_{i=1}^{K} \mathbb{E}[p_{l,i}]) \\ &+ \sum_{j=1}^{J} 3L^2 B^2 \sum_{l=j}^{J} \eta_l \sum_{t=j}^{l-1} \eta_t^2 s_t \sum_{k=1}^{t} (l - \tau_l) \alpha^{l+t-j-k} \|\nabla F(w_j)\|_2^2 \\ &+ \sum_{j=1}^{J} 3\eta_j L^2 \sigma_c^2 B^2 \sum_{l=1}^{j} \alpha^{j-l} \sum_{t=l}^{j-1} (j - \tau_l) \eta_t^2 s_t^2.\end{aligned} \tag{22}$$

Equation (22) amounts to:

$$\begin{aligned}&\frac{1}{J} \sum_{j=1}^{J} (\frac{\eta_j s_j}{2} - 3L^2 B^2 \sum_{l=j}^{J} \eta_l \sum_{t=j}^{l-1} \eta_t^2 s_t \sum_{k=1}^{t} (l - \tau_k) \alpha^{l+t-j-k}) ||\nabla F(w_j)||_2^2 \\ &\leq \frac{1}{J} \sum_{j=1}^{J} (\frac{3\eta_j s_j}{2} (G^2 + \sigma_e^2 \sum_{i=1}^{K} \mathbb{E}[p_{l,i}]) + 3\eta_j L^2 \sigma_c^2 B^2 \\ &\sum_{l=1}^{j} \alpha^{j-l} \sum_{t=l}^{j-1} (j - \tau_l) \eta_t^2 s_t^2) + \frac{F(w_1) - F(w^*)}{J}.\end{aligned}$$

B.3 Proof of Theorem 2

According to Lipschitz continues and Lemma 1 in FLDetector [Zhang *et al.*, 2022], we have

$$\begin{aligned}&\mathbb{E}d_{t,i} - \mathbb{E}d_a \\ =&\mathbb{E}\|\hat{g}(w_{t,i}, \zeta_{t,i})) + \hat{H}^t (w_t - w_{t-1}) - \hat{g}(w_{t-1,i}, \zeta_{t-1,i}))\| \\ &- \mathbb{E}\|\hat{g}(w_{t,a}, \zeta_{t,a})) + \hat{H}^t (w_t - w_{t-1}) - \hat{g}(w_{t-1,a}, \zeta_{t-1,a}))\| \\ \geq& 2\mathbb{E}\|\nabla f(w_t, \mathcal{D}_i)\| - 2(N+2)L\eta \|\nabla f(w_{t-1}, \mathcal{D}_i)\| \\ =&(2 - 2(N+2)L\alpha)\mathbb{E}\|\nabla f(D_j, w_{t-1})\| \geq 0\end{aligned}$$

C Experiments

Sparse Model Settings. The sparse model settings are customized for each client based on the width of their model d_i, which follows a normal distribution with mean m and standard deviation 0.05. Four mask levels (mask1, mask2, mask3, mask4) represent sparse models with 20%, 50%, 75% and 100% of global model parameters. Specifically, $d_i \leq 0.25$ corresponds to mask1, $d_i \in (0.25, 0.5]$ to mask2, $d_i \in (0.5, 0.75]$ to mask3, and $d_i \geq 0.75$ to mask4.

Attack Settings. Following prior work, we select 100 clients for training, with 28% randomly designated as malicious by default. The attack scenarios include label flipping (LF) attacks, backdoor attacks, and trim attacks.

Compared Methods. We compare asynchronous federated learning methods (TWAFL, SASGD), defense methods against Byzantine attacks (FoolsGold, FLDetector), and recovery methods (Retrain, FedEraser, Crab).

Datasets. Experiments are conducted on image classification tasks using MNIST, Fashion-MNIST, and CIFAR-10.

Experimental Equipment. All clients and the server are simulated on a workstation with a 2.4GHz Intel Core i9-12900 processor, NVIDIA RTX A5000 GPU, and 64GB memory.

Evaluation Metrics. Detection Accuracy (DACC), Recall Rate (RR), and Detection Precision Rate (DPR) measure the correctness, completeness, and precision of malicious client identification. Test Accuracy (TACC) evaluates model performance, Attack Success Rate (ASR) measures backdoor effectiveness, and Time Consumption tracks recovery duration.

FL Setting. The mask level is set to 0.2 and 0.5 to represent different distributions of the sparse client model. The Dirichlet distribution parameter is 0.8 for Non-IID data, and the staleness level is configured as $N/K = 1000/20$ for asynchronous scenarios.

Detector Setting. With staleness $N/K = 50/10$ and mask level $m = 0.5$, the experiment involves 50 clients (10 malicious). Attacks include trim and backdoor, with detection starting at the 50th training round and ensuring at least one attacker per round.

Recover Setups. The stagnation is set to $N/K = 50/20$ and the mask level is set to $m = 0.7$. The experiment uses 50 clients with 5, 10, or 20 malicious clients launching trim or backdoor attacks. Detection occurs in the 50th round and recovery is compared in the 60th round.

Acknowledgments

This work is supported by National Natural Science Foundation of China under grants 62171132, 62471139, and U1905211, and Natural Science Foundation of Fujian Province under grant 2024J09032.

References

[Bagdasaryan *et al.*, 2020] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, pages 2938–2948, 2020.

[Blanchard *et al.*, 2017] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 118–128, 2017.

[Cao *et al.*, 2021] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *Proceedings of NDSS*, 2021.

[Cao *et al.*, 2023] Xiaoyu Cao, Jinyuan Jia, Zaixi Zhang, and Neil Zhenqiang Gong. Fedrecover: Recovering from poisoning attacks in federated learning using historical information. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1366–1383, 2023.

[Chen *et al.*, 2017] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, volume 1, pages 1–25, 2017.

[Fang *et al.*, 2020] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *Proceedings of the 29th USENIX Conference on Security Symposium*, pages 1605–1622, 2020.

[Fung *et al.*, 2020] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pages 301–316, 2020.

[Hong *et al.*, 2022] Junyuan Hong, Haotao Wang, Zhangyang Wang, and Jiayu Zhou. Efficient splitmix federated learning for on-demand and in-situ customization. In *the 20th International Conference on Learning Representations*, 2022.

[Huang *et al.*, 2023] Siquan Huang, Yijiang Li, Chong Chen, Leyu Shi, and Ying Gao. Multi-metrics adaptively identifies backdoors in federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4652–4662, 2023.

[Jiang *et al.*, 2025] Yu Jiang, Jiyuan Shen, Ziyao Liu, Chee Wei Tan, and Kwok-Yan Lam. Towards efficient and certified recovery from poisoning attacks in federated learning. *IEEE Transactions on Information Forensics and Security*, 20:2632–2647, 2025.

[Konečný *et al.*, 2017] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv:1610.05492*, 2017.

[Liu and Nocedal, 1989] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528, 1989.

[Liu *et al.*, 2021] Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pages 1–10, 2021.

[Nguyen *et al.*, 2022] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider. Flame: Taming backdoors in federated learning. In *the 31st USENIX Security Symposium*, pages 1415–1432, 2022.

[Sun *et al.*, 2023] Jingwei Sun, Ang Li, Lin Duan, Samiul Alam, Xuliang Deng, Xin Guo, Haiming Wang, Maria Gorlatova, Mi Zhang, Hai Li, and Yiran Chen. Fedsea: A semi-asynchronous federated learning framework for extremely heterogeneous devices. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, page 106–119, 2023.

[Wang *et al.*, 2023] Yangyang Wang, Xiao Zhang, Mingyi Li, Tian Lan, Huashan Chen, Hui Xiong, Xiuzhen Cheng, and Dongxiao Yu. Theoretical convergence guaranteed resource-adaptive federated learning with mixed heterogeneity. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 2444–2455, 2023.

[Xia *et al.*, 2019] Qi Xia, Zeyi Tao, Zijiang Hao, and Qun Li. Faba: an algorithm for fast aggregation against byzantine attacks in distributed neural networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4824–4830, 2019.

[Xie *et al.*, 2021] Chulin Xie, Minghao Chen, Pin-Yu Chen, and Bo Li. Crfl: Certifiably robust federated learning against backdoor attacks. In *Proceedings of the 38th International Conference on Machine Learning*, pages 11372–11382, 2021.

[Zhang *et al.*, 2022] Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 2545–2555, 2022.