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

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 outcomes, FedHAN exhibits enhanced reliability and robustness in intricate and dynamic federated learning scenarios.

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.*, 2020a; Bagdasaryan *et al.*, 2020; Fang *et al.*, 2020a; Fang *et al.*, 2020b], 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.*, 2020b; 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 solu-

tion involves adjusting model parameters and adding Gaussian noise [Xie *et al.*, 2021; Nguyen and 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.*, 2024] 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 of this work include:

- **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:** In our preliminary experiments, we observed that some clients participating in asynchronous federated learning introduced errors and, in the absence of an efficient model recovery mechanism, accuracy decreased. Motivated by these findings, we devised an innovative integration of the aforementioned techniques. Specifically, the server keeps a cache of past client updates, which are utilized to complete missing parameters in sparse updates. These imputed updates are assigned adaptive weights depending on their staleness, and consistent historical updates are chosen to tackle Non-IID data challenges. By examining discrepancies between local and global updates, we identify malicious attackers. Upon their removal, the model is reconstructed using the Cauchy mean value theorem.
- **Convergence Proven, Suspicion Quantified for Secure Federated Learning:** We present a mathematical demonstration of the convergence upper bound for the FedHAN algorithm, along with the determination of a suspicion score threshold that separates benign from malicious clients. This results in two primary conclusions: (1) Both the staleness and the size of the model greatly influence the speed of convergence; (2) The suspicion scores calculated by FedHAN successfully discriminate between attackers and benign clients.
- **Elevating Accuracy and Resilience Against Federated Adversaries:** We performed a comprehensive series of tests across various settings. The findings indicate that FedHAN enhanced model accuracy by 7.86% in comparison with the most recent federated learning algorithms. Furthermore, in trials that involved three different poisoning attack scenarios, including backdoor attacks, FedHAN's detection and recovery algorithms surpassed the leading methods by 12% and 7.16%, respectively. These results affirm the robustness of the FedHAN algorithm in real-world and intricate federated learning contexts.

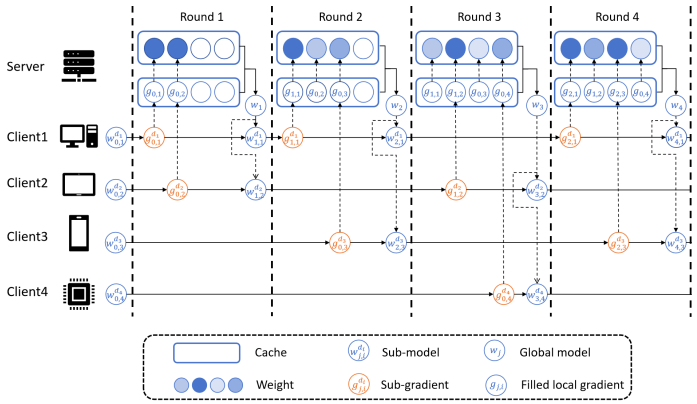


Figure 1: Cache-Based FedHAN Model Update Mechanism Architecture

2 Background and Related Work

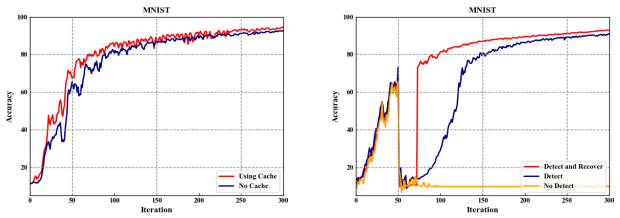
2.1 Poisoning Attacks in SAFL

Federated learning is vulnerable to attacks that skew local data or model updates, like backdoor [Bagdasaryan *et al.*, 2020] and trim [Fang *et al.*, 2020a] attacks, impacting model accuracy and reliability. 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.*, 2020b], 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.

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.*, 2024] 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. (A detailed background and related work description is in Appendix A.)



(a) (b)

Figure 2: Motivation experiments: (a) Evaluated the effect of incorporating historical updates on model training. (b) Assessed the influence of recovery algorithms on model training following attack detection.

3 Methodology

3.1 Problem Framework

We explore a federated learning framework that protects a central server and N clients. Initially, each client submits their model width d_i, and the server generates a tailored initial model $w_{0,i}^{d_i}$ 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 server. The server keeps a local update cache of size N, storing the latest updates from each client. An aggregation round is initiated after the cache is updated K times, leading to an update of the global model:

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

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. (Model Width) [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, where $d \in (0, 1]$. Specifically, when $d = 1$, the size of w^d is equivalent to the size of the global model.

Definition 2. (Customized sparse model) [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_i} denote the customized sparse model for client i, d_i represent the model width of client i, and m^{d_i} denote the model shape corresponding to d_i. According to Definition 2, the sparse model for a client can be calculated as follows:

$$w^{d_i} = w \odot m^{d_i}. \tag{2}$$

Appendix B provides a summary of the notations used in this paper.

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 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.

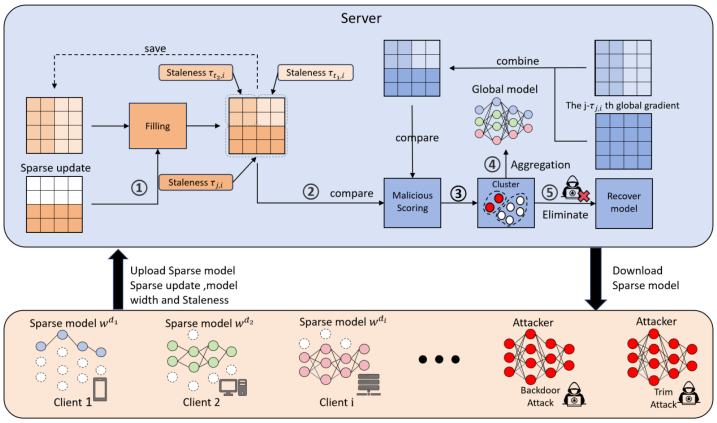


Figure 3: Update Imputation, Attack Detection, and Recovery Architecture in FedHAN

3.4 Overall Design of FedHAN

The FedHAN system is composed of three main modules: caching and model aggregation, attack detection, and model recovery. This modular structure allows the system to handle challenges efficiently and safely. Algorithm 1 details the FedHAN execution through these steps: (1) Sparse Model Customization: The server generates a client-specific mask m^{d_i} based on client capacity, customizes a sparse model $w^{d_i} = w \odot m^{d_i}$, and sends it to the client. (2) Local Training and Upload: Clients train their sparse models on local data and upload the sparse update $g(w_{t,i}^{d_i}, \xi_{t,i})$ 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 $\bar{g}(w_{t,i}, \xi_{t,i})$ (see Equation 3), forms a weight matrix based on update timing and client masks (see Equation 4), and updates caches.

Algorithm 1 FedHAN

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

Server side:

Initialize model parameter w_0 and iteration $j = 1$
Clients send model width d_i to server.
Server customizes $w_j^{d_i} = w_j \odot m^{d_i}$ to all clients.
while the stopping criteria is not satisfied **do**
    $stage = 1$
    $\bar{g}(w_{j,i}) \leftarrow FAA(\cdot)$
    $\tilde{g}(w_{j,i}) \leftarrow SHGAA(\cdot)$
    $g(w_j) = \bar{g}(w_{j,i}) + \alpha\tilde{g}(w_{j,i})$
    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

Client side:

Receive model $w_t^{d_i}$ from server at round t.
Perform update descent to get $g(w_t^{d_i}, \xi_{t,i})$ based on samples $\xi_{t,i}$.
Upload $g(w_t^{d_i}, \xi_{t,i})$ and model widthd_i.

(4) Malicious Update Detection: The server identifies malicious clients using Algorithm 5 by checking the Euclidean distance $s_{j,i} = \|\tilde{g}(w_{j,i}) - \bar{g}(w_{j,i})\|_2$ 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 help from benign clients and historical data (Algorithm 4). (6) Update Aggregation: Updates from K clients are aggregated to form $\bar{g}(w_j)$ (Algorithm 2). The server picks some unreceived updates from cache based on cosine similarity to form $\tilde{g}(w_j)$ (see Algorithm 3). (7) Global Model Update: Finally, the server computes and applies the global update using Equation 7. Appendix C provides detailed descriptions of Algorithms 2, 3, 4, and 5.

3.5 Implementation Details of FedHAN

3.5.1 Sparse Model Imputation

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, $g(w_{j,i}^d)$, reaches the server, i.e., $i \in C_j$, it will be incorporated into the historical updates, specifically $g(w_{j,i}^d, \xi_{j,i}) \cup \bar{g}(w_{j-1,i}) \odot \tilde{m}^{d_i}$; Otherwise, the latest imputed update will be the imputed update from the most recent round.

$$\bar{g}(w_{j,i}) = \begin{cases} g(w_{t,i}^d, \xi_{j,i}) \cup \bar{g}(w_{j-1,i}) \odot \tilde{m}^{d_i} & \text{if } i \in C_j \\ \bar{g}(w_{j-1,i}) & \text{if } i \in C_j^- \end{cases} \tag{3}$$

where, $g(w_{t,i}^d, \xi_{j,i})$ 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.

$$M_{j,i} = \begin{cases} M'_{j,i}(\beta^{\tau_{j,i}} m^{d_i} \cup \beta\tilde{m}^{d_i}) & \text{if } i \in C_j \\ \beta M_{j-1,i} & \text{if } i \in C_j^- \end{cases} \tag{4}$$

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

We assign a weight to the sparse model update $M'_{j,i}$ uploaded by the client, as it directly participates in the model update, while the remaining part $\alpha M'_{j,i}$ is assigned a separate weight. The calculation is as follows:

$$p'_{j,i} = M'_{j-1,i}(m^d \cup \alpha\tilde{m}^{d_i}) \tag{5}$$

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

3.5.2 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, and $\bar{g}(w_j)$ represents the aggregated result of the K most recent updates received by the server during the j-th iteration.

3.5.3 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 $s_{j,i} = \|\check{g}(w_{j,i}) - \bar{g}(w_{j,i})\|_2$, 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 higher average suspicious score are identified as malicious clients and removed from the current round. This process is summarized in Algorithm 5.

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 $\check{g}(w_{j,i})$ is updated synchronously with the latest imputed sparse update. Once the latest imputed sparse update $\bar{g}(w_{j,i})$ is imputed with $g(w^d_{t,i}, \xi_{j,i})$, the latest imputed global update $\check{g}(w_{j,i})$ is updated according to the following rule:

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

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

3.5.4 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 FedHIR protocol and estimated 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. This innovative design balanced estimated updates and precise computations, minimizing error risks and ensuring stable and reliable global model recovery.

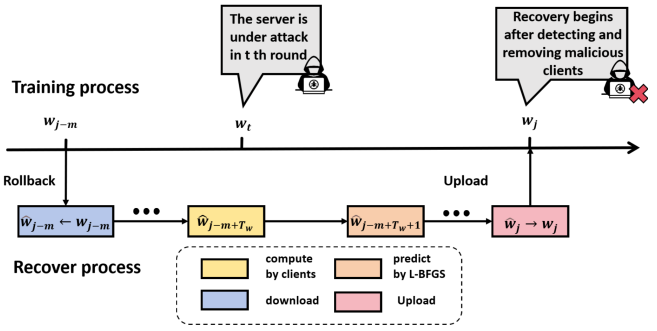


Figure 4: Model Recovery Process Details

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 $\Delta g^i_t = \bar{g}(\hat{w}_{t,i}, \xi_{t,i}) - \bar{g}(w_{t,i}, \xi_{t,i})$, 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, $\Delta G^i_t = [\Delta g^i_{b_1}, \Delta g^i_{b_2}, \ldots, \Delta g^i_{b_s}]$, 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^i_t 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. Additionally, the recovery algorithm requires the 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 needs to select historical updates based on cosine similarity. Assuming the number of clients satisfying $\tau_{j,i} \leq \tau_{max}$ 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 can be expressed as: $O(T_w + \frac{m-T_w}{T_c}s)$, where, T_w represents the number of rounds requiring precise updates at the early stage of the recovery process, $\frac{m-T_w}{T_c}$ represents the number of rounds requiring periodic precise updates during recovery, and s denotes the number of rounds needed for the recovery process. Thus, the total additional communication cost is: $O(T_w + \frac{m-T_w}{T_c}s)$.

4.2 Convergence Analysis

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

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

Then, we can obtain the following convergence result

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

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 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 have 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:*

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

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 E.

5 Experiments and Discussion

5.1 Experimental Setup

We assess FedHAN with three image classification datasets: MNIST, FMNIST, and CIFAR-10. We tested three attacks: backdoor, trim, and label-flipping. We compare asynchronous federated learning methods like TWAFL and SASGD; defense methods against Byzantine attacks such as FoolsGold and FLDetector; and recovery strategies such as Retrain, FedEraser, and Crab. The experiments include 2000 communication rounds, each with one epoch of local training and a batch size of 4, using an SGD optimizer with a learning rate of 0.005. The model has two convolutional layers and two fully connected layers. Data heterogeneity is emulated with Dirichlet distribution values of 0.3 and 0.8, while model heterogeneity uses mask levels of 0.2 and 0.5. The momentum α is set at 0.2, and the decay rate β at 0.9. More details of the experimental setup are listed in Appendix D.

5.2 Comparison of FedHAR with Other Models

Results and Analysis: Figures 5 highlight FedHAN's superior performance on MNIST, Fashion-MNIST, and CIFAR-10 datasets, compared to TWAFL and GASGD, with faster accuracy improvements and smoother training curves, showcasing its stability. FedHAN's performance remains robust across varying staleness levels ($N/K = 1000/10$ vs. $N/K = 1000/20$), as its client selection mechanism mitigates errors caused by client non-participation. Although TWAFL effectively balances old and new update information, the experimental data (Table 1) confirm the superiority of FedHAN in accuracy, convergence speed, and robustness, addressing heterogeneity and Non-IID challenges effectively. Its innovative design enhances model generalization and maintains high accuracy across diverse data distributions, proving valuable for real-world federated learning applications.

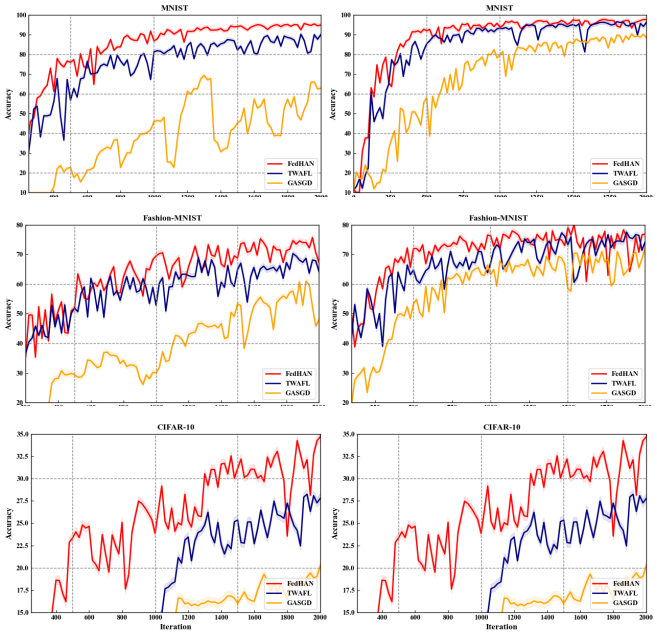


Figure 5: The performance comparison of three different algorithms under $N/K = 1000/10$ and $1000/20$ on the MNIST, FMNIST and CIFAR-10 dataset.

Dataset	mask level	Dirichlet distribution	N/K=1000/10			N/K=1000/20		
			FedHAN	TWAFL	SASGD	FedHAN	TWAFL	SASGD
MNIST	$m=0.5$	0.8	**97.63**	93.52	82.63	98.91	95.37	90.14
		0.3	**95,84**	90.93	76.29	97.13	93.47	90.25
	$m=0.2$	0.8	**93.95**	89.31	66.34	97.77	94.87	89.54
		0.3	**91.62**	85.11	60.34	95.19	91.05	84.25
Fashion EMNIST	$m=0.5$	0.8	**78.37**	75.31	61.80	82.93	78.45	65.75
		0.3	**76.61**	71.27	54.62	80.41	73.31	52.05
	$m=0.2$	0.8	**74.63**	71.51	54.35	77.44	74.67	63.74
		0.3	**71,84**	67.93	49.29	74.62	70.37	58.63
CIFAR-10	$m=0.5$	0.8	**45.24**	38.24	31.24	47.31	41.42	31.51
		0.3	**41.04**	33.72	25.32	45.16	37.89	25.05
	$m=0.2$	0.8	**36.02**	28.16	27.63	45.87	37.98	27.76
		0.3	**33.02**	23.46	21.63	43.94	34.49	21.94

Table 1: Prediction accuracy of FedHAN under varying levels of staleness, model heterogeneity, and Non-IID degree.

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 and 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	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 EMNIST	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 Results of FedHAN Compared to Other Algorithms.

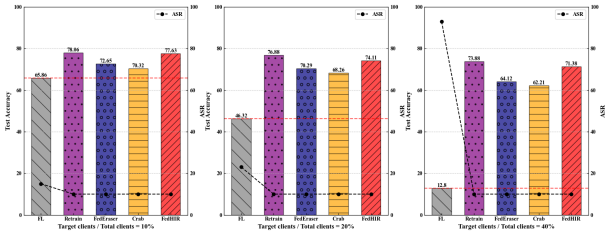


Figure 6: The test accuracy and ASR on MNIST dataset after recovering from backdoor attack when attack intensity is at 10%, 20% and 40% respectively.

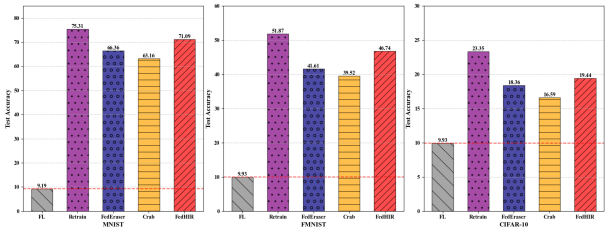


Figure 7: The test accuracy on MNIST, Fashion-MNIST and CIFAR-10 after recovering from trim attack.

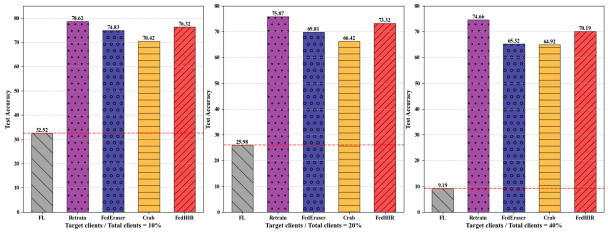


Figure 8: The test accuracy on MNIST dataset after recovering from trim attack when attack intensity is at 10%, 20% and 40% respectively.

5.4 Ablation Study of the Recovery Algorithm

Test Data Accuracy: Figures 6, 7, and 8 demonstrate that in analyzing the MNIST dataset at the 60th recovery round, increasing malicious client percentages caused a significant drop in test accuracy (e.g., 12.8% with 40% malicious clients in backdoor attacks). However, the FedHIR algorithm consistently beat other recovery methods such as FedEraser and Crab, showing its strength in tackling attacks and boosting model resilience.

Attack Success Rate (ASR): Figure 9 illustrates that the Attack Success Rate of the Recovery Algorithm (ASR) dropped to approximately 10%, 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 the MNIST dataset, underscoring the system's capability in preserving data privacy and model security.

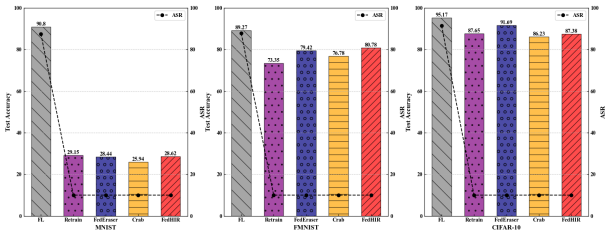


Figure 9: The MISR (presented in the bar chart) and ASR (presented in the line chart) on MNIST, Fashion-MNIST and CIFAR-10 after recovering from backdoor attack.

5.5 Discussion

Experimental results and theoretical analysis indicate that FedHAN successfully reduces the effects of outdated updates and Non-IID data on the global model using update imputation, adaptive weighting, and consistent historical update selection. FedHAN utilizes past data to supplement updates and modifies the impact of each client's update according to its staleness, allowing delayed updates to meaningfully contribute to the global model. Moreover, by aggregating consistent updates based on cosine similarity, FedHAN ensures that the global update reflects diverse client contributions. Consequently, the accuracy of the model improves by 9. 56% in the default settings (see Table 1).

FedHAN efficiently detects malicious clients in federated environments with model heterogeneity and asynchronous communication by imputing updates and evaluating deviations between local and global updates. It uses clustering algorithms such as DBSCAN and k -means. Compared to traditional methods, FedHAN improves detection accuracy by 12% (see Table 2). Moreover, with imputed historical updates, FedHAN can reconstruct a robust global model after malicious client detection without a notable accuracy loss, outperforming state-of-the-art methods by 7.16% (see Figure 6).

6 CONCLUSION

We identify key challenges in federated learning under practical conditions, such as Non-IID data, asynchronicity, model heterogeneity, and poisoning attacks, and introduce FedHAN, a cache-based semi-asynchronous algorithm that boosts robustness. FedHAN combines sparse model filling with adaptive weight distribution and selective historical updates to counteract stale updates and Non-IID data. It maintains a padded global update while imputing client sparse updates, allowing for accurate detection of malicious attacks. Using the Cauchy mean value theorem, it eradicates their impact, ensuring model reliability. We validate the convergence and attack detection efficiency of FedHAN, with experiments on three datasets affirming its superior accuracy and robustness, highlighting its practical applicability.

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 Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 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*, Red Hook, NY, USA, 2017. Curran Associates Inc.

[Cao *et al.*, 2021] X. Cao, M. Fang, J. Liu, and N. 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. *Proc. ACM Meas. Anal. Comput. Syst.*, 2017.

[Chen *et al.*, 2020] Yang Chen, Xiaoyan Sun, and Yaochu Jin. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE Transactions on Neural Networks and Learning Systems*, 31(10):4229–4238, 2020.

[Diao *et al.*, 2021] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2021.

[Fang *et al.*, 2020a] 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*. USENIX Association, 2020.

[Fang *et al.*, 2020b] 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*, SEC'20, USA, 2020. USENIX Association.

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

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

[H. Xiao and Vollgraf, 2017] K. Rasul H. Xiao and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*, 2017.

[Hong *et al.*, 2022] Junyuan Hong, Haotao Wang, Zhangyang Wang, and Jiayu Zhou. Efficient split-mix federated learning for on-demand and in-situ customization. In *ICLR*, 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 *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4629–4639, 2023.

[Jhunjhunwala *et al.*, 2022] Divyansh Jhunjhunwala, Pranay Sharma, Aushim Nagarkatti, and Gauri Joshi. Fedvarp: Tackling the variance due to partial client participation in federated learning. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 906–916, 2022.

[Jiang *et al.*, 2019] Yuang Jiang, Shiqiang Wang, Bong Jun Ko, Wei-Han Lee, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *arXiv:1909.12326*, 2019.

[Jiang *et al.*, 2024] Yu Jiang, Jiyuan Shen, Ziyao Liu, Chee Wei Tan, and Kwok-Yan Lam. Towards efficient and certified recovery from poisoning attacks in federated learning. *arXiv:2401.08216*, 2024.

[Karimireddy *et al.*, 2020] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Scaffold: stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning*. JMLR.org, 2020.

[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.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/cifar.html, 2009.

[LeCun *et al.*, 1998] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. Mnist handwritten digit database. http://yann.lecun.com/exdb/mnist, 1998.

[Lin *et al.*, 2020] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2020.

[Liu and Nocedal, 1989] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(1–3):503–528, August 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 and et al., 2022] Thien Duc Nguyen and et al. Flame: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 222)*, 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*, 2023.

[Thapa *et al.*, 2022] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[Wang *et al.*, 2022] Haozhao Wang, Ruixuan Li, Chengjie Li, Pan Zhou, Yuhua Li, Wenchao Xu, and Song Guo. Gradient scheduling with global momentum for asynchronous federated learning in edge environment. *IEEE Internet of Things Journal*, 2022.

[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.

[Wu *et al.*, 2021] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. Safa: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, 2021.

[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*, 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*, 2021.

[Zhang *et al.*, 2016] Wei Zhang, Suyog Gupta, Xiangru Lian, and Ji Liu. Staleness-aware async-sgd for distributed deep learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2350–2356. AAAI Press, 2016.

[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 Knowledghong2022efficiente Discovery and Data Mining*, page 2545–2555, 2022.

[Zhao *et al.*, 2018] Yang Zhao, Martin Li, Li Lai, Naveen Suda, Daniel Civin, and Virginia Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

A Background and Related Work

A.1 Heterogeneous Device

In Federated Learning (FL), traditional algorithms (such as Federated Averaging) require all clients to run models with the same size and structure. However, this requirement is difficult to meet on resource-constrained Internet of Things (IoT) devices. These devices often have limited computational power and storage capacity, making it challenging to support complex models. Therefore, a key challenge in FL research is how to adapt to the resource limitations of different clients while maintaining model performance.

To address this, researchers have proposed various flexible model aggregation methods. For example, FedDistill [Lin *et al.*, 2020] utilizes ensemble distillation techniques to aggregate local models of the same type into teacher models, while the server model acts as a student model, updated using KL divergence loss. However, this method relies on additional unlabeled data. HeteroFL [Diao *et al.*, 2021] assigns global sparse models to clients based on their resources, scaling input and output channels proportionally to adapt to resource constraints, although it may suffer performance degradation due to reduced training samples. Other methods include selective parameter pruning in Prune FL [Jiang *et al.*, 2019], model decomposition and recombination in Split-Mix [Hong *et al.*, 2022], split learning in SplitFed [Thapa *et al.*, 2022], and model customization in RAM-Fed [Wang *et al.*, 2023]. In general, these studies highlight the importance of developing flexible and efficient methods to accommodate heterogeneous client resources while ensuring effective training and maintaining model performance.

A.2 Semi-Asynchronous

Most Federated Learning (FL) algorithms, such as FedAvg, adopt the Synchronous Federated Learning (SFL) framework, where the server waits for all clients to complete local training and upload updates in each communication round before aggregating them. However, in heterogeneous device environments, differences in computational resources and network bandwidth between clients lead to inconsistent training times, causing delays in the server receiving updates and slowing down the overall training process. This limitation has driven the development of Asynchronous Federated Learning (AFL) and Semi-Asynchronous Federated Learning (SAFL). In AFL, the server updates the global model immediately after receiving updates from any client, but this frequent communication increases system overhead. In contrast, SAFL aggregates updates from a subset of clients in the order they arrive without waiting for all devices to complete training, balancing between SFL and AFL. However, stale updates in SAFL can reduce training efficiency and model accuracy.

To address these issues, researchers have proposed several improvements, such as assigning aggregation weights inversely proportional to update staleness [Chen *et al.*, 2020; Zhang *et al.*, 2016], selectively aggregating or discarding stale updates based on client state evaluation [Wu *et al.*, 2021], and ensuring update consistency with adaptive learning rates in frameworks such as WKAFL [Chen *et al.*, 2020], aiming to enhance the efficiency and performance of SAFL.

A.3 Non-IID Data

In Federated Learning (FL), Non-IID data refer to significant statistical differences between client data sets, which can negatively impact model convergence and accuracy, especially in highly skewed data sets [Zhao *et al.*, 2018]. In semi-asynchronous communication, where only a subset of clients participates in model aggregation, this issue is further exacerbated, leading to global models biased toward the data of participating clients.

To address this, several methods have been proposed. The SCAFFOLD algorithm [Karimireddy *et al.*, 2020] uses variance reduction to correct discrepancies between server and client updates, but maintaining control variates is a challenge for resources-constrained devices. Wang et al. proposed a global momentum strategy [Wang *et al.*, 2022] that accumulates historical information to mitigate biases caused by Non-IID data. Similarly, the FedVARP strategy [Jhunjhunwala *et al.*, 2022] by Jhunjhunwala et al. stores and reuses the most recent updates of non-participating clients on the server but does not account for update staleness. Zhang et al. extended this with RAM-Fed [Wang *et al.*, 2023], which stores the latest updates for all clients in heterogeneous model scenarios, but it still ignores the impact of stale update. Building on these methods, FedHAN selectively integrates updates from non-participating clients, retaining only consistent updates and assigning weights based on their staleness. This approach effectively leverages non-participating client information while mitigating the impact of update staleness, significantly enhancing model robustness, accuracy, and convergence speed.

A.4 Poisoning Attack in SAFL

Federated Learning (FL), as a distributed machine learning paradigm, faces threats from malicious attacks. Malicious clients can compromise the system by manipulating local training data or manipulating model updates (e.g., backdoor attacks [Bagdasaryan *et al.*, 2020] and trim attacks [Fang *et al.*, 2020a]). These tampered updates undermine the accuracy and reliability of the global model, posing significant risks to the security of FL systems.

To address this issue, researchers have proposed various malicious client detection methods that take advantage of differences in the feature distribution between benign and malicious updates. For example, the FABA algorithm [Xia *et al.*, 2019] identifies potential malicious clients by marking updates that deviate the most from the average update. The FoolsGold method [Fung *et al.*, 2020b] detects malicious behavior by analyzing the similarity between clients, while FLDetector [Zhang *et al.*, 2022] uses the Cauchy mean value theorem to compare predicted and actual updates, identifying clients with significant deviations.

However, in scenarios that involve model heterogeneity and asynchronous communication, the effectiveness of existing methods is limited. In asynchronous FL, stale updates can deviate significantly from the global update, making it harder to distinguish between stale and malicious updates. Additionally, model heterogeneity causes substantial differences in update distributions between clients, further increasing complexity. FedHAN addresses these challenges by intro-

ducing caching and model imputation mechanisms, partially mitigating the issues arising from asynchronous communication and model heterogeneity. This approach enhances the robustness and detection capabilities of FL systems in complex scenarios.

A.5 Machine Unlearning

Existing methods to detect malicious clients often require multiple rounds of detection, during which the global model may already be compromised. Mitigating the impact of malicious clients after detection thus becomes a critical challenge. A simple solution is to retrain the global model using only benign clients, but this incurs high communication costs.

Current recovery methods aim to reduce communication overhead by utilizing historical information stored during training. For example, FedRecovery [Cao *et al.*, 2023] predicts updates using the Cauchy mean value theorem, FedEraser [Liu *et al.*, 2021] calibrates updates with historical data, and Crab [Jiang *et al.*, 2024] selects key historical information based on KL divergence and cosine similarity. However, these methods assume that malicious clients have already been detected and lack integration with mainstream attack detection approaches. Moreover, their effectiveness in real-world federated learning scenarios is limited.

To address these challenges, FedHAN introduces the first integrated algorithm that combines detection and recovery for Byzantine attacks in federated learning. This unified approach adapts to heterogeneous environments, significantly improving system robustness and practicality in complex scenarios.

B Notations

Notation	Description
N	total number of clients
C_j	the set of selected clients
C_R	the set of remain clients
T	total Number of Communication Rounds
d	model width
m^{d_i}	mask of Client i's Model
$\tilde{m}^{d}$	Inverse mask of Client i's Model
$\odot$	Customizing operation
$\cup$	Imputing sparse model operation
w_j	original global model in round j
$\tilde{w}_t$	recovered global model in round t
$g(w_{t,i}^{d_i}, \xi_{t,i})$	lobal sparse update of client i
$\bar{g}(w_{j,i})$	latest imputed update
$\bar{g}(w_j)$	estimated unbiased update

Table 3: Notations

C Algorithms

Algorithm 2 Sparse Model Imputation $(\cdot)$

Require: historcal update $g(w_{j,c}^{d})$,last imputed update ,last weight matrix$M_{j-,i}$

for $i \in C_j$ **do**
  Receive next round model width , the update and its staleness $(d_i, g(w_{t,i}, \xi_{j,i}), \tau_i)$ from the i-th client
  $\bar{g}(w_{t,i}, \xi_{t,i}) = g(w_{t,i}^{d_i}, \xi_{t,i}) \cup \bar{g}(w_{j-1,i}) \odot \tilde{m}^{d_i}$
  $\bar{g}(w_{j,i}) = \bar{g}(w_{t,i}, \xi_{t,i})$
  $M'_{j,i} \leftarrow \beta^{\tau_{j,i}} M'_{j-1,i}(m^{d} \cup \beta \tilde{m}^{d_i})$
  $p'_{j,i} = M'_{j-1,i}(m^{d_i} \cup \alpha \tilde{m}^{d_i})$
  if $stage = 1$ **then**
    $mark[i][t] \leftarrow 1$
    $\bar{M}'_{t,i} = M'_{j,i} * \beta^{-\tau_{j,i}}$
    store $\bar{g}(w_{t,i}, \xi_{t,i})$ and $\bar{M}'_{t,i}$
  end if
end for
$p_{j,i} = p'_{j,i} / \sum_{i=1}^{K} p'_{j,i}$ $\bar{g}(w_j)$ according to Equation D
$\bar{g}(w_j) = \sum_{i=1}^{K} p_{j,i} \bar{g}(w_{j,i})$.
return $\bar{g}(w_j)$

Algorithm 3 Select historical updates and aggregate$(\cdot)$

Require: last imputed update $\bar{g}(w_{j,i})$, ,last weight matrix$M_{j,i}$

initial an empty set $\mathcal{T}$
for $i \in C_j^-$ **do**
  if $\tau_{j,i} \leq \tau_{max}$ **then**
    $sim_{j,i} = \cos\langle \bar{g}(w_{j,i}), \bar{g}(w_j)) \rangle$
    $\mathcal{T} \leftarrow \mathcal{T} \cup sim_{j,i}$
  end if
  $M'_{j,i} \leftarrow \beta M'_{j-1,i}$
end for
$\{\bar{g}(w_{j,i})\}_{i=1}^{X} \leftarrow \text{Select}(\mathcal{T}, \{\bar{g}(w_{j,i})\}_{i=1}^{|C_j^-|})$
$p_{j,i} = M'_{j,i} / \sum_{i=1}^{X} M'_{j,i}$
$\tilde{g}(w_j) = \sum_{i=1}^{X} p_{j,i} \bar{g}(w_{j,i})$.
return $\tilde{g}(w_j)$

Algorithm 4 Recovery Algorithm

Input: recover round T $\{w_j\}_{j=1}^{T}, \{\bar{g}(w_j)\}_{j=1}^{T}, \{\bar{g}(w_{j,i})\}_{c=1}^{X}$
Output: $\tilde{w}$

Initialize model parameter w_0 and iteration $j_0 = T - m$
Clients send model width d_i to server.
Server customizes $\hat{w}_j^{d_i} = \hat{w}_j \odot m^{d_i}$ to all clients.
for $j \leftarrow j_0$ to T **do**
    Obtain the set of indices where $mark[:, j] == 1$, denoted as $\hat{C}_j$
    if $j < j_0 + T_w$ *or* $(j - T_w)\ mod\ T_c == 0$ **then**
        $\bar{g}(\hat{w}_j) \leftarrow FAA(\cdot)$
    else
        for $i \in C_j$ **do**
            $\hat{H}_{j,i}(\hat{w}_j - w_j)$
            $=$ L-BFGS$(\Delta W_j, \Delta G_{j,i}, \hat{w}_j - w_j)$
            $\hat{g}(\hat{w}_{j,i}, \xi_{j,i}) = \bar{g}(w_{j,i}, \xi_{j,i}) + \hat{H}_{j,i}(\hat{w}_j - w_j)$
            $\bar{g}(\hat{w}_{j,i}) = \hat{g}(\hat{w}_{j,i}, \xi_{j,i})$
            $\hat{M}'_{j,i} = \bar{M}'_{j,i}$
            $\tilde{g}(\hat{w}_j) \leftarrow SHGAA(\cdot)$
            $g(\hat{w}_j) = \bar{g}(\hat{w}_j) + \alpha\tilde{g}(\hat{w}_j)$
            $\hat{w}_{j+1} \leftarrow \hat{w}_j - n_j g(\hat{w}_j)$
        end for
    end if
end for
return $\hat{w}$

Algorithm 5 Detect malicious clients

Input: mini-batch size m **output:** update $g(w_{j,i}^d, \xi_{j,i})$, delay

for all client $i = 1, 2, \ldots, K$ **do**
    for $i \in C_R$ **do**
        if $i \in C_j$ **then**
            $\check{g}(w_{j,i}) = g(w_{j-\tau_{j,i}}) \odot m_i^{d_i} \cup \check{g}(w_{j-1,i}) \odot \tilde{m}_i^{d_i}$
        else
            $\check{g}(w_{j,i}) = \check{g}(w_{j-1,i})$
        end if
        $s_{j,i} = \|\check{g}(w_{j,i}) - \bar{g}(w_{j,i})\|_2$
    end for
    append $s_{j,i}$ into S_j
end for
Call DBSCAN for pre-detection get k
if $k > 1$ **then**
    Perform k-means clustering based on $k = 2$
    Remove the clients in the cluster with larger average scores
end if
return None

D Experiments

Sparse Model Settings: sparse model settings are customized based on each client's model width d_i, assumed to follow a normal distribution with a 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 [Zhang *et al.*, 2022], we select 100 clients for training and, by default, randomly designate 28% of them as malicious clients. The attack scenarios include the following types: label flipping (LF) Attack [Cao *et al.*, 2021], backdoor Attack [Bagdasaryan *et al.*, 2020]and trim Attack [Fang *et al.*, 2020a]

Compared Methods: We compared different asynchronous federated learning methods, including TWAFL [Chen *et al.*, 2020] and SASGD [Zhang *et al.*, 2016]; various defense methods against Byzantine attacks, including FoolsGold [Fung *et al.*, 2020b]and FLDetector [Zhang *et al.*, 2022]; and different recovery methods, including Retrain, FedEraser [Liu *et al.*, 2021], and Crab [Jiang *et al.*, 2024].

Datasets: We considered multiple image classification datasets for various learning tasks, specifically including MNIST [LeCun *et al.*, 1998], Fashion-MNIST [H. Xiao and Vollgraf, 2017], and CIFAR-10 [Krizhevsky and Hinton, 2009].

Experimental Equipment: In our experiments, we simulated all clients and the server on a workstation. The workstation is equipped with a 2.4GHz Intel Core i9-12900 processor and an NVIDIA RTX A5000 graphics card. Additionally, the workstation has 64GB of memory. This configuration provides us with powerful computational resources that allow efficient simulation experiments for both clients and the server.

Evaluation Metrics: To evaluate detection and recovery performance, we use distinct metrics for each. (1) Detection Metrics: Detection Accuracy (DACC) measures the proportion of correctly classified clients, Recall Rate (RR) indicates the percentage of actual malicious clients correctly identified, and Detection Precision Rate (DPR) reflects the proportion of identified malicious clients that are truly malicious. Maximizing DACC, RR, and DPR ensures robust detection. (2) Recovery Metrics: Test Accuracy (TACC) assesses the global model's prediction accuracy, Attack Success Rate (ASR) measures the model's misclassification rate for backdoor-triggered samples, and Time Consumption tracks the duration of each recovery round.

FL Setting: We set the mask level to 0.2 and 0.5, representing two distributions of client sparse model sizes. The Dirichlet distribution parameter is set to 0.8, representing different degrees of Non-IID data. Additionally, we configure two different staleness levels, $N/K = 1000/20$ and $N/K = 1000/10$, to simulate varying degrees of asynchronous communication scenarios.

Detector Setting: We set the staleness level to $N/K = 50/10$ and the mask level to $m = 0.5$. The experiment includes a total of 50 clients, with 10 malicious clients. The attackers employ three different attack methods: trim attack, backdoor attack, and label-flipping attack. Detection is performed from the 50th training round. We ensure that at least one attacker participates in each training round.

Recover Setups: We set the staleness parameter at $N/K = 50/20$ and the mask level at $m = 0.7$. In the experiment, there are a total of 50 clients and the number of malicious clients is set to 5, 10, and 20, respectively. The attackers employ two different attack methods: the trim at-

tack and the backdoor attack. We perform attack detection in the 50th training round and compare the results using the recovery algorithm in the 60th round.

E Convergence Analysis

E.1 Assumption

Assumption 1. *Lipschitz Continuity. Objective function F(·) 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* $\nabla F_i(w_j)$*:*

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

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 the 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 $\nabla F(w_l)$.

E.2 Proof 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

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

Define $s_j = \sum_{l=1}^{j} \alpha^{j-l} = \frac{1-\alpha^j}{1-\alpha}$and take expectation on both sides of Equation (24):

$$\begin{aligned}
&\mathbb{E}[F(w_{j+1})] - F(w_j) \\
&= -\frac{\eta_j}{2} \sum_{l=1}^{j} \alpha^{j-l}\mathbb{E}[\sum_{i=1}^{\mathrm{R}} p_{l,i}(||\nabla F(w_j)||_2^2 \\
&\quad + ||\bar{g}(w_{l,i}, \xi_{l,i})||_2^2 - ||\nabla F(w_j) - \bar{g}(w_{l,i}, \xi_{l,i})||_2^2)] \\
&\quad + \frac{L}{2}\eta_j^2 s_j^2\mathbb{E}[\|\sum_{l=1}^{j} \frac{\alpha^{j-l}}{s_j} \sum_{i=1}^{K} p_{l,i}\bar{g}(w_{l,i}, \xi_{l,i})\|_2^2] \\
&\leq -\frac{\eta_j}{2} \sum_{l=1}^{j} \alpha^{j-l}\mathbb{E}[\sum_{i=1}^{K} p_{l,i}(||\nabla F(w_j)||_2^2 + ||\bar{g}(w_{l,i}, \xi_{l,i})||_2^2 \\
&\quad - \|\nabla F(w_j) - \bar{g}(w_{l,i}, \xi_{l,i})\|_2^2)] \\
&\quad + \frac{L}{2}\eta_j^2 s_j \sum_{l=1}^{j} \alpha^{j-l}\mathbb{E}[||\sum_{i=1}^{K} p_{l,i}\bar{g}(w_{l,i}, \xi_{l,i})||_2^2] \qquad (16) \\
&\leq -\frac{\eta_j}{2} \sum_{l=1}^{j} \alpha^{j-l}\mathbb{E}[\sum_{i=1}^{K} p_{l,i}(\|\nabla F(w_j)\|_2^2 + ||\bar{g}(w_{l,i}, \xi_{l,i})||_2^2 \\
&\quad - ||\nabla F(w_j) - \bar{g}(w_{l,i}, \xi_{l,i})||_2^2)] \\
&\quad + \frac{L}{2}\eta_j^2 s_j \sum_{l=1}^{j} \alpha^{j-l}\mathbb{E}[\sum_{i=1}^{K} p_{l,i}||\bar{g}(w_{l,i}, \xi_{l,i})||_2^2] \qquad (17)
\end{aligned}$$

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

Equations (27) and (28) are derived on the basis of Jensen's inequality. Equation (29) is derived because the learning rate satisfies $\eta_j \leq \frac{1}{Ls_j}$. With respect to term $\mathcal{A}$,

$$
\begin{aligned}
\mathcal{A} &= \mathbb{E}[\sum_{i=1}^{K} p_{l,i}\|\nabla F(w_j) - \nabla F_i(w_j) + \nabla F_i(w_j) \\
&\quad - \nabla F_i(w_{\tau(l)}) + \nabla F_i(w_{\tau(l)}) - \bar{g}(w_{l,i},\xi_{l,i})\|_2^2] \\
&\leq 3\mathbb{E}[\sum_{i=1}^{K} p_{l,i}\|\nabla F(w_j) - \nabla F_i(w_j)\|_2^2] \\
&\quad + 3\mathbb{E}[\sum_{i=1}^{K} p_{l,i}\|\nabla F_i(w_j) - \nabla F_i(w_{\tau(l)})\|_2^2] \\
&\quad + 3\mathbb{E}[\sum_{i=1}^{K} p_{l,i}\|\nabla F_i(w_{\tau(l)}) - \bar{g}(w_{l,i},\xi_{l,i})\|_2^2] \\
&\leq 3G^2 + 3L^2\underbrace{\mathbb{E}[\sum_{i=1}^{K} p_{l,i}\|w_j - w_{l-\tau_{l,i}}\|_2^2]}_{\mathcal{B}} + 3\sigma_e^2\sum_{i=1}^{K}\mathbb{E}[p_{l,i}];
\end{aligned} \qquad (24)
$$

Equation (24) is derived based on Assumptions 3.1 and 3.3. Define $\tau_l = \max(1, l - \tau_{\max})$. With respect to term $\mathcal{B}$,

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

Equation (25) is derived from Lemma B.1. By replacing Equations (24), (25) into Equation (23), we have

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

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

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

Equation (32) amounts to:

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

E.3 Proof Theorem 2

$$
\begin{aligned}
&\mathbb{E} d_i - \mathbb{E} d_a \\
=&\mathbb{E} \|\hat{g}(w_{t,i}, \zeta_{t,i})) + \hat{H}^t (w_t - w_{t-1}) - \hat{g}(w_{t-1,i}, \zeta_{t-1,i}))\| \\
&- \mathbb{E} \|\hat{g}(w_{t,a}, \zeta_{t,a})) + \hat{H}^t (w_t - w_{t-1}) - \hat{g}(w_{t-1,a}, \zeta_{t-1,a}))\| \\
=&\mathbb{E} \|\nabla f(w_t, \mathcal{D}_i) + \hat{H}^t (w_t - w_{t-1}) - \nabla f(w_{t-1}, \mathcal{D}_i)\| \\
&- \mathbb{E} \|\nabla f(w_t, \mathcal{D}_a) + \hat{H}^t (w_t - w_{t-1}) - \nabla f(w_{t-1}, \mathcal{D}_a)\|
\end{aligned}
$$

According to Lipschitz continues we have:

$$
\begin{aligned}
\geq& 2\mathbb{E}[\|\nabla f(w_t, \mathcal{D}_i)\| - L\|w_t - w_{t-1}\| - \|\hat{H}^t (w_t - w_{t-1})\|] \\
&- \mathbb{E}[(\|\nabla f(w_t, \mathcal{D}_a) - \nabla f(w_{t-1}, \mathcal{D}_a)\| + \|\hat{H}^t (w_t - w_{t-1})\|)] \\
\geq& 2\mathbb{E}\|\nabla f(w_t, \mathcal{D}_i)\| - 2(L\|w_t - w_{t-1}\| + \|\hat{H}^t (w_t - w_{t-1})\|)
\end{aligned}
$$

According to Lemma 1 in FLDetector [13], we have:

$$
\begin{aligned}
&\geq 2\mathbb{E}\|\nabla f(w_t, \mathcal{D}_i)\| - 2(N+2)L\|w_{l+1} - w_l\| \\
&\geq 2\mathbb{E}\|\nabla f(w_t, \mathcal{D}_i)\| - 2(N+2)L\eta\|\nabla f(w_{t-1}, \mathcal{D}_i)\| \\
&= (2 - 2(N+2)L\alpha)\mathbb{E}\|\nabla f(D_j, w_{t-1})\| \\
&\geq 0
\end{aligned}
$$