SKYLAB - UC BERKELEY

Research Internship

April 2024 - August 2024

Saidane Ayoub
Cohort 2021
8th company
Football Section

ÉCOLE POLYTECHNIQUE

INSTITUT POLYTECHNIQUE DE PARIS

sky UC Berkeley

Declaration of Integrity Regarding Plagiarism

I, the undersigned, hereby declare that the work I am submitting is my own and that I have not engaged in any form of plagiarism. I have properly cited and referenced all sources of information and ideas that are not originally mine, in accordance with the academic integrity policies of this institution.
I understand that plagiarism is a serious academic offense and that any violation of this declaration may result in disciplinary action, including the possibility of academic penalties.

23/08/2024
Ayoub Saidane

Contents

1 Introduction **4**
- 1.1 Background . . . 4
- 1.2 Objectives . . . 5
  - 1.2.1 Evaluating the Performance of Robust Aggregation Rules . . . 5
  - 1.2.2 Implementing and Testing Sequential Training Orchestration . . . 6

2 Literature Review **6**
- 2.1 Federated Learning . . . 6
  - 2.1.1 Basics of Federated Learning . . . 6
- 2.2 Key Challenges and Recent Advancements . . . 7
- 2.3 Byzantine Robustness . . . 9
  - 2.3.1 Overview of Byzantine Faults and Their Impact on Federated Learning . . . 9
  - 2.3.2 Review of Existing Robust Aggregation Rules . . . 10
- 2.4 Sequential Training in Federated Learning . . . 11

3 Contributions **12**
- 3.1 Robust Aggregation . . . 12
  - 3.1.1 Aggregation rules . . . 12
  - 3.1.2 Byzantine Attack Models . . . 13
  - 3.1.3 Experimental Setup . . . 14
  - 3.1.4 Experimental results: . . . 15
- 3.2 Sequential Training . . . 18
  - 3.2.1 Sequential Algorithm . . . 18
  - 3.2.2 Experimental Setup . . . 20
  - 3.2.3 Experimental Results . . . 22

4 Conclusion **27**

Enhancing Federated Learning with Robust Aggregation and Sequential Orchestration

Abstract

This report examines the effectiveness of Federated Learning (FL) in mitigating Byzantine attacks within environments where clients have heterogeneous data distributions. The study compares the robustness of different aggregation methods and introduces sequential training as an alternative to traditional parallel training approaches. The primary goal is to improve the Byzantine resilience and accuracy of FL systems while effectively managing the challenges posed by data heterogeneity across participating clients.

1 Introduction

1.1 Background

Overview of Federated Learning (FL)

Federated Learning (FL) represents a decentralized paradigm in machine learning, allowing multiple clients—whether individual devices or entire organizations—to collaboratively train a shared model without the need for direct data exchange [1–3]. This decentralized approach inherently safeguards data privacy by keeping data localized to each client, which is particularly crucial in sectors where data sensitivity is paramount, such as healthcare, finance, and mobile applications. The capacity of FL to maintain data confidentiality while enabling collaborative learning has driven its adoption across a broad range of applications.

Despite its advantages, FL is not without significant challenges. These include issues related to device and data heterogeneity, communication overhead, and security vulnerabilities [4–8]. Device and data heterogeneity refers to the variations in computational power, storage capacity, connectivity, and data distribution among clients, which can create inefficiencies and disparities in the learning process. Communication overhead arises from the frequent exchange of model updates between clients and the central server, which can strain network bandwidth and increase latency. Furthermore, security concerns, particularly those related to Byzantine attacks, pose a significant threat, as malicious clients can potentially disrupt the training process or compromise the integrity of the global model [2, 9].

Ongoing research is actively addressing these challenges, to enhance FL's robustness and scalability, thereby solidifying its role as a key technology for privacy-preserving, scalable, and personalized machine learning.

Challenges related to Byzantine Resilience in FL

Byzantine resilience is a critical aspect of Federated Learning (FL), ensuring that the system maintains its integrity and functionality even in the presence of malicious or faulty participants. Given the inherently distributed nature of FL, trust among participants cannot always be guaranteed, making the system susceptible to Byzantine attacks that can disrupt the training process, degrade model performance, or even cause system failure. To mitigate these risks, it is essential to implement robust aggregation methods and secure protocols that can effectively filter out or minimize the impact of malicious updates. These techniques play a vital role in maintaining the accuracy and reliability of the global model, despite the presence of adversarial behaviors within the network [10–12].

Challenges related to Heterogeneous Data Distributions

Heterogeneous data distribution, or non-independent and identically distributed (non-IID) data, presents a significant challenge in FL. Clients generate and store data reflecting their specific usage patterns, leading to diverse and uneven data distributions across the network. This heterogeneity can cause local models to diverge, impairing the performance and convergence of the global model. Various approaches, such as the introduction of proximal terms in algorithms like FedProx [13] and clustered FL, have been proposed to address this challenge and enhance the robustness and applicability of FL in real-world scenarios [1, 14].

1.2 Objectives

In this report, we primarily evaluate the performance of several promising aggregation rules tested against potential Byzantine attacks in Federated Learning. Additionally, we address the challenging issue of robustness in heterogeneous data distributions by implementing a sequentially orchestrated algorithm. Understanding these aspects is critical for securely deploying FL in real-world scenarios where data heterogeneity and potential adversarial behavior are inevitable.

1.2.1 Evaluating the Performance of Robust Aggregation Rules

The primary objective of this study is to evaluate the effectiveness of various robust aggregation rules in enhancing the reliability and accuracy of FL systems. These aggregation rules filter out malicious or corrupted updates from participating clients, ensuring that the model converges to an accurate and robust state [15].

This involves developing and evaluating aggregation methods that can effectively handle and mitigate the impact of Byzantine failures. The goal is to ensure that the aggregated model updates are accurate and reliable, even in the presence of malicious or faulty client contributions. Techniques such as Coordinate-wise Median (CwMed), Coordinate-wise Trimmed Mean (CwTM), Mean around Median (MeaMed), Comparative Gradient Elimination (CGE), and Centered Clipping (CC) will be explored to enhance the robustness of the aggregation process [9, 10, 12, 15].

1.2.2 Implementing and Testing Sequential Training Orchestration

Another key objective is to implement and evaluate sequential training orchestration in FL environments. Sequential training organizes the training process in a specific order, rather than simultaneously, to improve efficiency and convergence rates. This method also aims to enhance the algorithm's robustness by reducing the influence of Byzantine nodes during the learning process. The implementation will be tested across various configurations and datasets to assess its impact on training performance and model accuracy [4, 5].

This area focuses on sequentially orchestrating the training process to improve efficiency and model robustness [4, 5]. By processing client updates sequentially, the goal is to mitigate the influence of Byzantine nodes in the learning process and aggregate only the models that demonstrate the best performance. The sequential training approach will be implemented and tested in various FL settings where the clients' dataset distribution can be either homogeneous or heterogeneous.

2 Literature Review

2.1 Federated Learning

2.1.1 Basics of Federated Learning

The FL process is characterized by iterative training and communication rounds, wherein clients perform local computations and share model updates rather than raw data with the central server.

A pivotal component of FL is the aggregation method employed by the central server to integrate the diverse updates from the clients. One of the most widely adopted aggregation techniques is Federated Averaging (FedAvg) [16]. FedAvg operates by averaging the local model updates from each participating client, weighted by the size of their local datasets. This strategy ensures that the global model reflects the contributions of clients proportionally to their data volume, thereby accommodating heterogeneity in data distributions and client participation. The advantages of FedAvg include its simplicity, scalability, and effectiveness in handling non-IID (non-Independent and Identically Distributed) data settings, which are prevalent in real-world FL applications. The overall training process utilizing FedAvg is delineated in Algorithm 1.

Algorithm 1: Parallel Federated Learning (FedAvg)

Input: Initial global model parameters $\mathbf{w}^0$

Output: Trained global model parameters $\mathbf{w}^T$

Initialization: The central server initializes the global model parameters $\mathbf{w}^0$ and distributes them to a selected subset $\mathcal{S}_t$ of K clients out of N total clients for each round t.

for *each round* $t = 1, \ldots, T$ **do**

  for *each client* $k \in \mathcal{S}_t$ ***in parallel*** **do**

    Local Training:

      Initialize $\mathbf{w}_k^t = \mathbf{w}^t$

      for *each local epoch* $e = 1, \ldots, E$ **do**

        for *batch* b *in client* k*'s data* **do**

          Update model parameters:

$$\mathbf{w}_k^t \leftarrow \mathbf{w}_k^t - \eta \nabla \ell(\mathbf{w}_k^t; b)$$

          where η is the learning rate and ℓ is the loss function.

    Send updated model parameters $\mathbf{w}_k^t$ to the central server.

  Aggregation (FedAvg):

  The central server updates the global model by computing a weighted average of the received client updates:

$$\mathbf{w}^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{w}_k^t$$

  where n_k is the number of data points at client k and $n = \sum_{k=1}^{K} n_k$ is the total number of data points from the selected clients in round t.

*Algorithm 1: Overview of the Federated Learning Process.

2.2 Key Challenges and Recent Advancements

Recent advancements in Federated Learning (FL) have addressed key challenges across several domains, including privacy, security, robustness, and efficiency. Differential Privacy and Secure Multi-Party Computation have enhanced data protection, while Byzantine-Resilient Aggregation techniques have improved system robustness against adversarial attacks. To tackle data and system heterogeneity, methods like FedProx and asynchronous FL allow for effective model training despite varying client data distributions and computational resources. Communication overhead has been mitigated through techniques like update compression, and advanced architectures like FATE-LLM have facilitated the federated training of large models. Additionally, model personalization strategies and incentive mechanisms have been developed to ensure that FL systems remain relevant, accurate, and collaborative, addressing the unique needs and motivations of participating clients.

Privacy and Security

Privacy and security are paramount concerns in Federated Learning, and recent advancements have made significant strides in addressing these issues. Differential Privacy (DP)

has emerged as a crucial technique, adding noise to model updates to protect individual data points from inference attacks. This ensures that even if an adversary gains access to the aggregated model, they cannot extract sensitive information about any specific client. Additionally, Secure Multi-Party Computation (SMPC) techniques, including homomorphic encryption and secret sharing, have been developed to secure the computation of updates. These methods protect the data during transmission, ensuring that the central server cannot directly access individual updates, thus maintaining the confidentiality of client data [2].

Robustness and Fault Tolerance

Robustness and fault tolerance are critical for the reliability of Federated Learning systems, especially in the presence of malicious clients. Recent methods, such as Byzantine-Resilient Aggregation, have been proposed to enhance the resilience of FL against adversarial attacks. Techniques like Krum, Trimmed Mean, and Median aggregation are designed to tolerate a certain proportion of malicious updates, ensuring that the integrity of the global model is maintained. Additionally, Adaptive Federated Learning strategies dynamically adjust aggregation rules or client participation based on varying levels of client reliability and data quality. This adaptability allows the system to respond effectively to fluctuations in the participating clients' performance and data distribution [10–12].

Data and System Heterogeneity

Data heterogeneity presents another significant challenge, as clients often possess non-independent and identically distributed (non-IID) data. This can lead to model divergence and reduced performance. To address this issue, approaches like FedProx have been introduced, which incorporate proximal terms to regularize local updates. Additionally, clustered Federated Learning groups clients with similar data distributions, facilitating the training of more effective models by leveraging the shared characteristics of their data [1].

System heterogeneity, characterized by clients with varying computational resources, poses challenges for fair participation and efficient training. Asynchronous Federated Learning has emerged as a solution, allowing clients to operate at their own pace without being synchronized with others. Furthermore, techniques like federated distillation enable lightweight clients to contribute by distilling knowledge from larger models, ensuring that all clients can participate meaningfully in the training process [7].

Communication Overhead

One of the most pressing challenges in Federated Learning is communication overhead. Frequent communication between clients and the central server can lead to high latency and bandwidth consumption, especially when dealing with large models. To mitigate this issue, techniques such as federated dropout, communication-efficient aggregation, and local update compression (including quantization and sparsification) have been employed. These strategies aim to reduce the amount of data that needs to be transmitted, thereby lowering communication costs and improving overall system efficiency [4].

Model Personalization

Another notable advancement in Federated Learning is the focus on model personalization. Federated Multi-Task Learning (FMTL) allows for the personalization of models for different clients by sharing common representations while simultaneously learning task-specific components. This approach enhances the relevance and accuracy of the models for individual clients, accommodating their unique data characteristics. Additionally, Clustered Federated Learning groups clients with similar data distributions, enabling the training of more specialized models. This clustering approach not only improves model performance but also reduces the risk of model divergence caused by heterogeneous data [3].

2.3 Byzantine Robustness

2.3.1 Overview of Byzantine Faults and Their Impact on Federated Learning

Byzantine faults refer to arbitrary and potentially malicious failures in a distributed system. Unlike benign faults, which are typically due to crashes or network issues, Byzantine faults can involve nodes (clients) behaving maliciously, sending corrupted or misleading information, or even colluding with other malicious nodes to disrupt the system.

Examples of Famous Attacks

- **Data Poisoning Attack:** Data poisoning involves injecting malicious data into the training dataset. In federated learning, an adversary can modify the local dataset on their device to contain incorrect or misleading data. When this corrupted data is used to train the model, it can significantly degrade the model's performance or introduce specific vulnerabilities that the attacker can exploit [10].

- **Model Poisoning Attack:** Similar to data poisoning, model poisoning directly targets the model updates. Here, a malicious client sends crafted updates that are designed to degrade the global model's performance or to bias the model in a particular way. This can be done subtly over several communication rounds to avoid detection [8].

- **Evasion Attack:** In this attack, the adversary generates inputs that are specifically designed to be misclassified by the trained model. These inputs are crafted based on the knowledge of the model's parameters or structure and are intended to fool the model into making incorrect predictions.

Impact on Federated Learning

In the context of Federated Learning (FL), Byzantine faults can significantly compromise the training process and the quality of the resulting model. Malicious clients may send incorrect model updates, leading to model degradation where the global model's performance deteriorates or becomes entirely ineffective. Such faults can also cause instability during training, resulting in fluctuations that prevent the model from converging. Additionally, Byzantine faults can manifest as data poisoning, where malicious clients introduce tainted data that biases or harms the model's predictions. Furthermore, colluding clients may exploit these faults to breach security measures, gaining unauthorized access

to data or model parameters of other clients.

Handling Byzantine failures is particularly challenging in the presence of heterogeneous data distributions across clients. The diversity in local datasets exacerbates the difficulty of distinguishing between legitimate updates and malicious ones, as the natural variations in data can mask or mimic the effects of Byzantine attacks. This heterogeneity can lead to situations where the global model becomes highly sensitive to the malicious updates, further destabilizing the learning process and making it harder to achieve a robust and reliable model. Consequently, ensuring Byzantine fault tolerance in such scenarios is not only crucial but also more complex.

2.3.2 Review of Existing Robust Aggregation Rules

Robust aggregation rules are essential in Federated Learning (FL) to mitigate the impact of Byzantine faults, ensuring that global model updates remain accurate and reliable even in the presence of malicious clients. These aggregation methods help maintain the integrity and robustness of the learning process by addressing potential adversarial attacks.

- **Krum** is a robust aggregation method that selects a single update from the clients, which is closest to the majority of other updates. This method calculates the Euclidean distance between each client's update and those from other clients, selecting the update with the smallest sum of distances to its closest neighbors. Krum is effective in filtering out extreme outliers and malicious updates, making it a reliable choice for maintaining model integrity. However, it is computationally expensive due to the numerous distance calculations required and is not scalable to large numbers of clients [17].

- **Trimmed Mean:** The Trimmed Mean method removes a fixed proportion of the highest and lowest values in each dimension of the updates and averages the remaining values. This approach discards the most extreme values for each dimension of the update vectors, computing the mean of what remains. Its simplicity and effectiveness in reducing the impact of outliers make it an attractive option. However, the choice of the proportion to trim is critical and may need to be adjusted depending on the level of Byzantine faults present [18].

- **Median:** Median aggregation involves taking the median value for each dimension of the updates. For each dimension, the median value across all client updates is computed and used to update the global model. This method is robust to extreme values and relatively easy to implement. Although it is computationally less efficient than mean-based methods, it offers greater resilience to outliers.

- **Bulyan:** Bulyan combines the strengths of Krum and Trimmed Mean to provide enhanced robustness. Initially, Krum is used to select a set of candidate updates, after which Trimmed Mean is applied to these candidates to compute the final aggregate update. This combination makes Bulyan highly robust to a wide range of Byzantine faults. However, it is computationally intensive due to the combined use of both Krum and Trimmed Mean [9].

- **Coordinate-wise Median and Trimmed Mean:** These methods apply median or trimmed mean aggregation in a coordinate-wise manner. For each dimension separately, the median or trimmed mean is computed, thus aggregating the updates. This approach maintains robustness while being computationally more feasible than some other methods. However, it may still be vulnerable if the proportion of Byzantine clients is too high [9].

These robust aggregation methods are crucial for enhancing the security and reliability of Federated Learning systems, particularly in environments where the risk of adversarial attacks is significant. Each method has its strengths and limitations, and the choice of which to implement may depend on the specific requirements and constraints of the FL application.

2.4 Sequential Training in Federated Learning

Sequential training, particularly with intelligent grouping as implemented in FedSeq, demonstrates significant potential in addressing the challenges of data heterogeneity in Federated Learning (FL). The integration of practical techniques with robust theoretical analysis establishes sequential training as a powerful method in FL, enhancing both the convergence speed and the overall robustness of models in diverse and realistic settings. The work presented in [4] introduces FedSeq, a novel framework specifically designed to handle the heterogeneity of data distributions across FL clients. The core innovation of FedSeq lies in its approach of grouping clients with diverse local datasets into "superclients" and conducting sequential training within these groups.

Clients are grouped into superclients to create larger, more homogeneous datasets, thereby reducing the negative impact of data heterogeneity on the global model. By performing sequential training within superclients, the model is exposed to a wider range of data before aggregation on the server, leading to improved convergence rates and enhanced model robustness. This approach is particularly effective for extreme heterogeneity, as it simulates training on a more uniform dataset, which mitigates the noise and divergence typically caused by non-i.i.d. data, ultimately accelerating convergence and enhancing model performance.

However, despite the progress made with FedSeq, there is currently no state-of-the-art implementation that leverages sequential training to ensure Byzantine resilience in FL. This report will focus on addressing this gap by exploring the potential of sequential training for enhancing Byzantine fault tolerance in federated learning systems.

3 Contributions

In this section, we present model aggregation methods designed to defend against Byzantine adversaries while adapting to heterogeneous data distributions across clients. Additionally, we implement sequential training algorithms that highlight the potential for optimizing model performance in both homogeneous and heterogeneous data settings.

3.1 Robust Aggregation

3.1.1 Aggregation rules

The following subsection describes various methods for aggregating the model weights received from all participating clients. Let n be the number of participating clients, f be the fraction of byzantine clients and $w_1, \ldots, w_n$ be the list of model weights, where each $w_i \in R^d$, and d is the dimension of the model weights. We use the same notation introduced in [9].

Centered clipping (CC)

Upon choosing a *clipping parameter* $c \geq 0$, we compute a sequence of model weights $v_0, \ldots, v_M$ in R^d such that for all $m \in [M]$:

$$v_m \leftarrow v_{m-1} + \frac{1}{n} \sum_{i \in [n]} (w_i - v_{m-1}) \min \left\{ 1, \frac{c}{\|w_i - v_{m-1}\|} \right\}$$

where M represents the number of iterations v_0 may be chosen arbitrarily. Then, $\mathrm{CC}(w_1, \ldots, w_n) = v_M$.

Comparative gradient elimination (CGE)

Let p denote a permutation on $[1, \ldots, n]$ that sorts the input list of model weights based on their norm in non-decreasing order. Then CGE outputs the average $n-f$ first weights:

$$\mathrm{CGE}(w_1, \ldots, w_n) = \frac{1}{n-f} \sum_{i=1}^{n-f} w_{p(i)}$$

Mean around median (MeaMed)

MeaMed computes the average of the $n - f$ closest elements to the median coordinate-wise. Specifically, for each $k \in [d], m \in [n]$, let $i_{m,k}$ be the index of the model weight with k-th coordinate that is the m-th closest to $\mathrm{Median}([w_1]_k, \ldots, [w_n]_k)$. Let C_k be the set of $n - f$ indices defined as

$$C_k = \{i_{1,k}, \ldots, i_{n-f,k}\}.$$

Then we have

$$[\mathrm{MeaMed}(w_1, \ldots, w_n)]_k = \frac{1}{n-f} \sum_{i \in C_k} [w_i]_k$$

Coordinate-wise trimmed mean (CwTM)

We denote by p_k the permutation on $[1, \ldots, n]$ that sorts the k-coordinate of the input list of model weights in non-decreasing order, i.e.,

$$[w_{p_k(1)}]_k \leq [w_{p_k(2)}]_k \leq \cdots \leq [w_{p_k(n)}]_k.$$

Then, $\mathrm{CwTM}(w_1, \ldots, w_n)$, is a vector in R^d is defined coordinate-wise as follows

$$[\mathrm{CwTM}(w_1, \ldots, w_n)]_k := \frac{1}{n-2f} \sum_{j=f+1}^{n-f} [w_{p_k(j)}]_k.$$

Coordinate-wise trimmed median (CwMed)

We denote by $[w_i]_k$ the k-coordinate of the model weight w_i, then $\mathrm{CwMed}(w_1, \ldots, w_n)$, is defined coordinate-wise as follows

$$[\mathrm{CwMed}(w_1, \ldots, w_n)]_k := \mathrm{Median}([w_1]_k, ..., [w_n]_k).$$

Computation efficiencies

The computational complexities of the aggregation rules suggest that these methods are comparably efficient and scalable, making them suitable for large-scale federated learning scenarios despite the varying computational demands.

Aggregation Rule	**Computational Complexity**
Coordinate-wise Trimmed Mean (CwTM)	$O(n \cdot d \cdot \log n)$
Coordinate-wise Median (CwMed)	$O(n \cdot d)$
Centered Clipping (CC)	$O(M \cdot n \cdot d)$
Comparative Gradient Elimination (CGE)	$O(n \cdot (d + \log n))$
Mean around Median (MeaMed)	$O(n \cdot d \cdot \log n)$

Table 1: Aggregation Rules and Their Computational Complexities

3.1.2 Byzantine Attack Models

The following subsection describes three types of Byzantine attacks that can disrupt the federated learning process by introducing malicious behavior from a fraction of participating clients.

Sign Flipping Attack

In a Sign Flipping attack, Byzantine clients intentionally invert the signs of their model updates before sending them to the central server. Formally, a Byzantine client C_i modifies its training gradient update to the opposite sign, which, when aggregated, can significantly degrade the model's convergence and final performance.

Label Flipping Attack

A Label Flipping attack occurs when Byzantine clients intentionally corrupt their training data by flipping the labels of the training examples. For example, in a binary classification task, a client could flip all labels of class 0 to class 1 and vice versa. The model updates w_i generated from this corrupted data are then sent to the server, causing the global model to learn incorrect associations between features and labels, leading to poor generalization and degraded accuracy.

Gaussian Attack

In a Gaussian Attack, Byzantine clients add noise sampled from a Gaussian distribution to their model updates. Formally, a Byzantine client C_i perturbs its model weight vector as follows:

$$w_i' = w_i + \mathcal{N}(0, \sigma^2)$$

where $\mathcal{N}(0, \sigma^2)$ represents Gaussian noise with mean 0 and variance σ^2. This type of attack introduces randomness into the aggregation process, which can cause the global model to converge more slowly or become stuck in suboptimal solutions, especially when a significant fraction of clients participate in the attack.

3.1.3 Experimental Setup

To compare the performance of these aggregation rules, we implement an extension of FedScale, an open-source federated learning (FL) framework that facilitates scalable and realistic FL research. FedScale is particularly suitable for this study as it provides high-level APIs for implementing FL algorithms and offers extensive datasets and benchmarking capabilities that emulate real-world FL scenarios [19].

Datasets and Tasks The experiments are centered around the FEMNIST dataset, a federated version of the classic MNIST dataset, extended to include 62 classes (digits and both uppercase and lowercase letters). FEMNIST is designed to simulate a realistic FL scenario with non-IID data distribution, where each client's data corresponds to handwritten characters from a unique writer. This non-IID nature of the data makes it an ideal benchmark for evaluating the robustness of different aggregation strategies under heterogeneous conditions. The data distribution across clients, as shown in Figure 1, highlights the inherent heterogeneity in the dataset.

Model Architecture For the classification task, we utilize the ResNet-18 model, a widely recognized deep convolutional neural network (CNN) architecture. ResNet-18 is chosen for its balance between depth and computational efficiency, making it suitable for deployment in federated learning environments, especially when clients possess varying computational capabilities.

Training Configuration The training process is configured to run for 100 rounds, with an evaluation interval set to test the model on the testing set every 5 rounds. Each round involves 50 participants, and local training on each client is performed using a batch size of 20 over 5 local steps. The learning rate is set to 0.05. To manage the computational load, 2 data loaders are used per client process, and clients with less than 21 samples are filtered out.

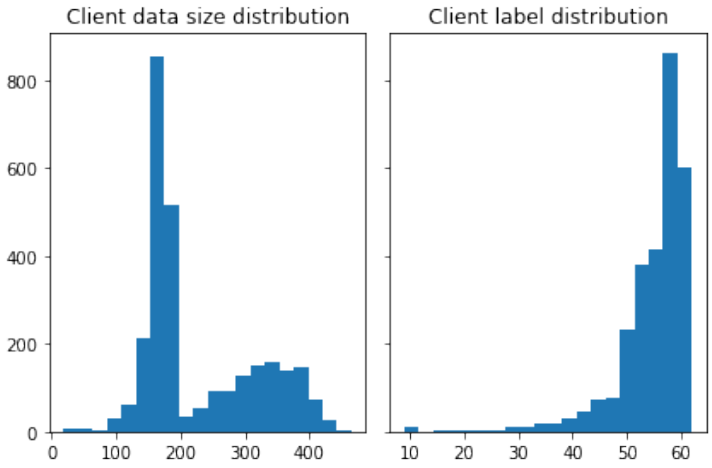
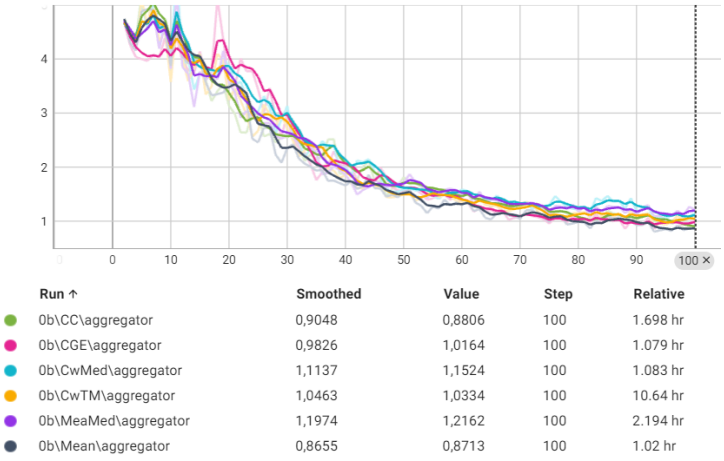


Figure 1: Data distribution across clients in the FEMNIST dataset.

Byzantine Configurations The Byzantine experiments include three types of attacks: Gaussian Attack, Label Flipping, and Sign Flipping. These attacks are applied by a fraction of the clients participating in the training process. The fractions of malicious clients tested are 0, 0.25, and 0.5.

Performance Metric In this experiment, the performance of the global model was evaluated using the CrossEntropy loss.

3.1.4 Experimental results:



Run ↑	Smoothed	Value	Step	Relative
0b\CC\aggregator	0,9048	0,8806	100	1.698 hr
0b\CGE\aggregator	0,9826	1,0164	100	1.079 hr
0b\CwMed\aggregator	1,1137	1,1524	100	1.083 hr
0b\CwTM\aggregator	1,0463	1,0334	100	10.64 hr
0b\MeaMed\aggregator	1,1974	1,2162	100	2.194 hr
0b\Mean\aggregator	0,8655	0,8713	100	1.02 hr

Figure 2: Performance of various aggregation methods in the absence of Byzantine attacks

Non Byzantine performance In the absence of Byzantine attacks (as shown in Figure 2), all aggregation methods demonstrated improved performance with smoother convergence patterns. Among them, the **Mean Aggregator** and **CC** emerged as the most effective, achieving the lowest final loss values. This highlights their efficiency and robustness in non-adversarial settings. While **MeaMed** and **CGE** were less efficient in

comparison, they still managed to converge to acceptable loss levels, indicating their capability to perform reasonably well even without the presence of Byzantine threats.

Byzantine Label Flipping (BLF) When the fraction of Byzantine clients is increased to 25%, as seen in Figure 3, the variance in the loss increases compared to the non Byzantine scenario. In this setup, **CC** continues to show strong performance, yielding the lowest smoothed loss. **CGE** and **CwMed** perform better than in the 50% Byzantine scenario, but they still lag behind **CwTM** and **CC** in overall effectiveness. In the presence of 50% Byzantine clients, as shown in Figure 3, the learning process exhibits significant instability, with a higher variance observed in the loss curves. Among the tested aggregation methods, **CGE** shows the highest loss, indicating its struggle under such high adversarial conditions. In contrast, **CwTM** performs the best, achieving the lowest final loss and demonstrating robustness against severe Byzantine interference. **CC** and the **Mean Aggregator** also display competitive performance, with relatively stable convergence.

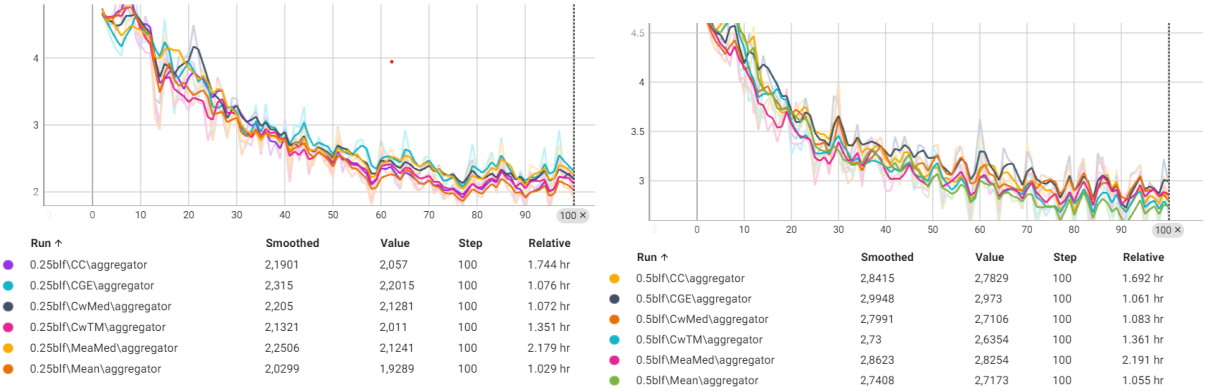


Figure 3: Performance of various aggregation methods in the presence of Label Flipping Byzantine attacks. The left image represents the 25% Byzantine Label Flipping scenario, and the right image represents the 50% Byzantine Label Flipping scenario.

Byzantine Sign Flipping (BSF) In the presence of 50% Byzantine Sign Flipping attacks, as shown in the right panel of Figure 4, all aggregation methods exhibit stable convergence with minimal variance. Despite the high proportion of Byzantine clients, methods like **CGE** and **CwTM** perform similarly, indicating a uniform impact across the board. The **Mean Aggregator** and **CC** (Centered Clipping) also demonstrate resilience, converging to similar loss values as other methods. Conversely, under the 25% BSF scenario (left panel of Figure 4), the learning process shows more variability, particularly with methods like **CwMed** and **MeaMed**, which struggle to achieve consistent convergence. In contrast, **CC** and **CGE** perform more robustly, with **CC** achieving the lowest smoothed loss, highlighting its effectiveness in mitigating the effects of BSF attacks at this lower fraction.

Gaussian Attack Analysis Under the 25% Byzantine Gaussian (BG) attack scenario, as shown in the left panel of Figure 5, the learning process shows varying degrees of instability across different aggregation methods. **CC** emerges as the most effective method, achieving the lowest smoothed loss, which underscores its robustness against moderate levels of Gaussian noise introduced by Byzantine clients. **CwMed** and **CwTM** also

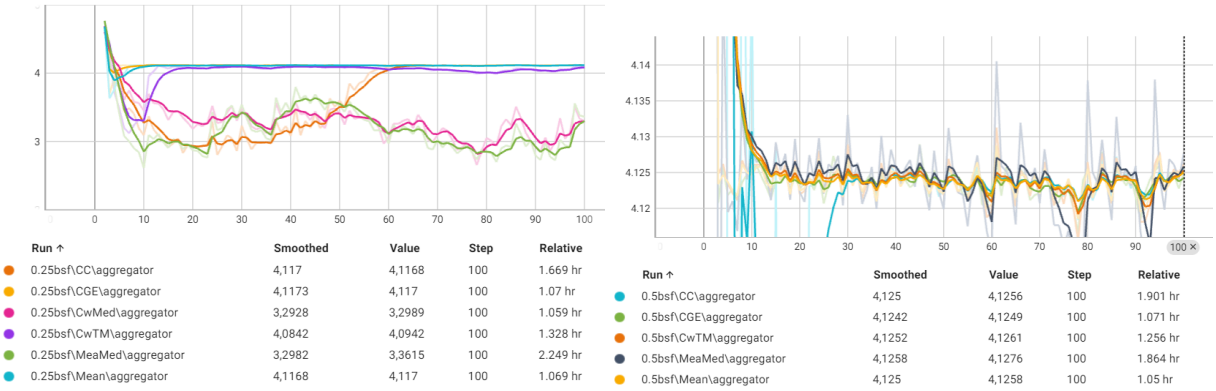


Figure 4: Performance of various aggregation methods in the presence of Sign Flipping Byzantine attacks. The left image represents the 25% Byzantine Sign Flipping scenario, and the right image represents the 50% Byzantine Sign Flipping scenario.

demonstrate decent performance, although they exhibit slightly higher losses than CC. **MeaMed** and **CGE** struggle to maintain consistent performance, with CGE particularly affected by the noise, leading to relatively high loss values. The **Mean Aggregator** is significantly impacted by the 25% Gaussian attack, showing poor convergence and high final loss values, indicating its vulnerability to this type of adversarial behavior.

When the proportion of Byzantine clients increases to 50%, as depicted in the right panel of Figure 5, the impact of the Gaussian attack becomes more pronounced across all methods. The **Mean Aggregator** exhibits the most substantial increase in loss, showing a sharp rise that highlights its severe vulnerability to such attacks. **CGE** and **MeaMed** also suffer significantly, with CGE reaching extremely high loss values, indicating its difficulty in coping with the substantial noise introduced by the 50% BG attack. On the other hand, **CC** continues to perform relatively well, maintaining the lowest loss among the methods, though with a slight increase compared to the 25% scenario. **CwMed** and **CwTM** exhibit similar trends as in the 25% scenario, with stable but slightly elevated loss values, further illustrating the robustness of these methods against Byzantine Gaussian attacks, even at higher adversarial fractions.

Results Analysis

The study of Byzantine-resilient federated learning primarily revolves around the development of robust aggregation methods that can mitigate the adverse effects of Byzantine nodes—nodes that may behave arbitrarily or maliciously during the training process. The methods explored, including Centered Clipping (CC), Coordinate-wise Trimmed Mean (CwTM), and others, address the critical need for robustness in distributed machine learning systems.

CC consistently emerged as one of the most resilient methods across various scenarios. Its ability to maintain stability even in the presence of a significant fraction of Byzantine nodes makes it a preferred choice for systems where robustness is paramount. The method's effectiveness stems from its capacity to reduce the influence of outliers by clipping the updates, thereby ensuring that the aggregated result remains close to the true average of the honest clients.

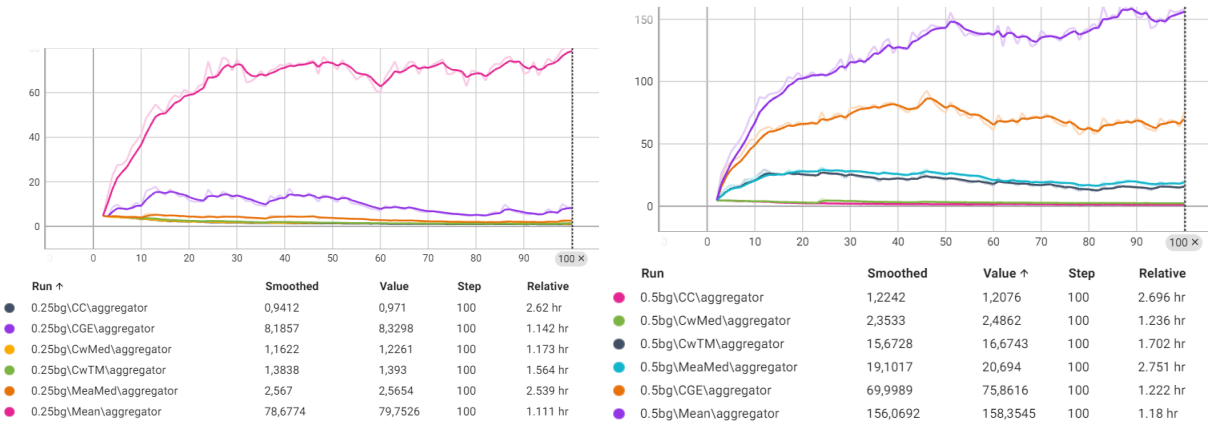


Figure 5: Performance of various aggregation methods in the presence of Gaussian Byzantine attacks. The left image represents the 25% Byzantine Gaussian scenario, and the right image represents the 50% Byzantine Gaussian scenario.

In contrast, **Mean Aggregator** and **Comparative Gradient Elimination (CGE)** struggled under adversarial conditions. These methods are more vulnerable to Byzantine attacks, especially when the fraction of Byzantine nodes increases. This vulnerability is primarily due to their reliance on simple averaging mechanisms, which are easily skewed by malicious updates.

Overall, the analysis indicates that selecting an appropriate aggregation method is crucial for maintaining both model performance and robustness in federated learning environments. **Centered Clipping** and **Coordinate-wise Trimmed Mean** are particularly reliable choices, offering a balance between stability and resilience against Byzantine behavior, while more straightforward aggregation methods like **Mean Aggregator** and **CGE** may not be suitable in high-risk scenarios.

3.2 Sequential Training

In this section, we implement a new federated learning algorithm based on sequential orchestration and compare its performance against parallel training across homogeneous and heterogeneous data distributions, as well as in the presence of malicious settings.

3.2.1 Sequential Algorithm

The first algorithm 2, **Fully Sequential Federated Training**, focuses on a purely sequential approach to federated learning. In each round, the server shuffles the clients and distributes the global model to the first client in the sequence. Each client receives the model updated by the previous client, trains it on its local data, and sends the updated parameters back to the server. This sequential process continues for all clients in the round. The fully sequential nature of this algorithm ensures that each client's contribution is fully integrated before the next client begins training. This method can be particularly useful in scenarios where the order of client updates significantly impacts the final model, although it may be slower compared to parallel approaches due to the sequential nature of updates.

The second algorithm 3, **Sequential Federated Learning**, describes a federated learning process that integrates a combination of parallel and sequential operations. Initially, the central server distributes the global model parameters to all participating clients. Each client trains its local model independently and sends the updated parameters back to the server. The server then clusters clients based on their model weights and forms sequences by grouping clients from different clusters. These sequences are further divided into blocks, and the server orchestrates the training of each block using different combinations of clients. The process is repeated R times for each combination, and the best-performing combination is selected for aggregation. Finally, the server aggregates the models from all sequences to update the global model 6. This approach allows for a more robust aggregation by considering various client combinations, especially in the presence of Byzantine clients.

Algorithm 2: Fully Sequential Federated Training

Input: Initial global model parameters $\mathbf{w}^0$
Output: Trained global model parameters $\mathbf{w}^T$
Initialization: The central server initializes the global model parameters $\mathbf{w}^0$.
for *each round* $t = 1, \ldots, T$ **do**
  The server shuffles the order of clients in $\mathcal{C}$ and distributes the global model to the first client
  foreach *client* $C_n \in \mathcal{C}$ ***sequentially*** **do**
    Client C_n receives from the server the model parameters $\mathbf{w}_{n-1}^t$ calculated by client C_{n-1}.
    Client C_n trains its local model $\mathbf{w}_n^t$ for several epochs.
    Client C_n sends the updated model parameters $\mathbf{w}_n^t$ to the server.

*Algorithm 2: Overview of the Federated Learning Process.

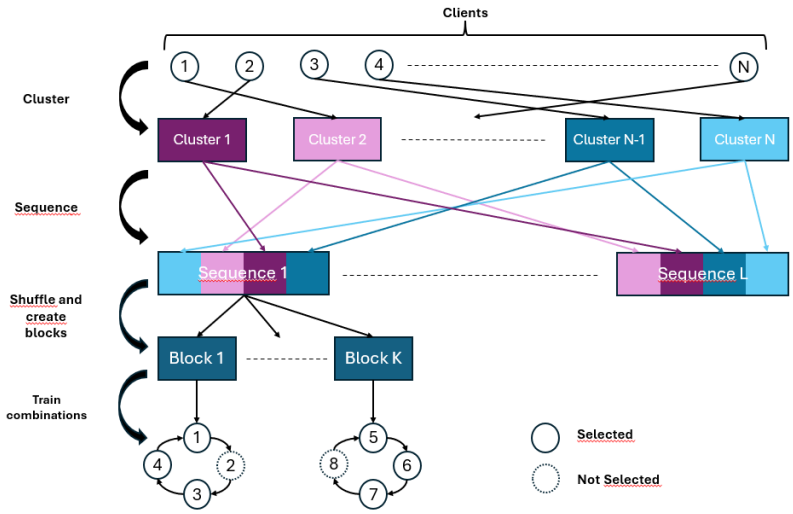


Figure 6: Illustration of the Sequential Federated Learning Process as described in Algorithm 3. This diagram shows the clustering of clients, the formation of sequences, the creation of blocks, and the training of combinations.

Algorithm 3: Sequential Federated Learning

Input: Initial global model parameters $\mathbf{w}^0$, Number of training repetitions R
Output: Trained global model parameters $\mathbf{w}^T$
Initialization: The central server initializes the global model parameters $\mathbf{w}^0$ and distributes them to all clients $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$.
for *each round* $t = 1, \ldots, T$ **do**
  foreach *client* $C_n \in \mathcal{C}$ ***in parallel*** **do**
    Client C_n trains its local model $\mathbf{w}_n^t$ for a number of epochs.
    Client C_n sends the local model update $\mathbf{w}_n^t$ back to the central server.
  Client Clustering and Sequence Formation:
    The server performs hierarchical clustering on the clients $\mathcal{C}$ to determine clusters $\mathcal{G} = \{G_1, G_2, \ldots, G_M\}$ based on the distance between their model weights $\mathbf{w}_n^t$.
    Create heterogeneous sequences $\mathcal{S} = \{S_1, S_2, \ldots, S_L\}$ by grouping clients from different clusters G_m together.
  Sequence Training:
    foreach *sequence* $\mathcal{S}_l \in \mathcal{S}$ **do**
      The server divides each sequence $\mathcal{S}_l$ into blocks $\mathcal{B}_l = \{B_{l1}, B_{l2}, \ldots, B_{lK}\}$ of a size determined by the length of the sequence multiplied by the fraction of Byzantine nodes.
      foreach *block* $B_{lb} \in \mathcal{B}_l$ ***in parallel*** **do**
        The server creates $|B_{lb}|$ different ordered combinations of clients $\mathcal{K}_{lb} = \{K_{lb1}, K_{lb2}, \ldots, K_{lbJ}\}$ in B_{lb} where one client is omitted in each combination.
        foreach *combination* $K_{lbj} \in \mathcal{K}_{lb}$ ***in parallel*** **do**
          repeat
            foreach *client* C_n *in combination* K_{lb} ***sequentially*** **do**
              Upon receiving from the server a model $\mathbf{w}_{n-1}^t$ trained on client C_{n-1}, client C_n trains it and sends the update back to the server.
          until R *times*;
        The server selects the best-performing combination based on accuracy.
      The server aggregates all block models
  Once all sequences $\mathcal{S}_l$ are trained, the server aggregates all sequence models to form the global model $\mathbf{w}^{\mathbf{t}}$.

*Algorithm 3: Overview of the Sequential Federated Learning Process.

3.2.2 Experimental Setup

Data Distribution In this experiment, we trained a ResNet-18 model on the FEMNIST dataset, which was partitioned in either a homogenous or heterogeneous distribution across clients, where the distribution of labels allocated to each client follows a Gaussian distribution of the data labels. This approach simulates a realistic federated learning scenario where clients have non-IID (non-Independent and Identically Distributed) data. Specifically, the labels allocated to each client were generated using the following procedure:

- For each client C_n, a mean label μ_n was determined by dividing the label space into equal intervals and assigning the center of each interval as the mean.
- The standard deviation σ was kept constant across all clients, ensuring that each client had a label distribution that is primarily centered around its mean μ_n but overlaps with adjacent clients. We test three different values for σ to compare the performance of the training processes.
- The label distribution for each client was generated based on a Gaussian probability density function, which determines the likelihood of each label being selected for the client.

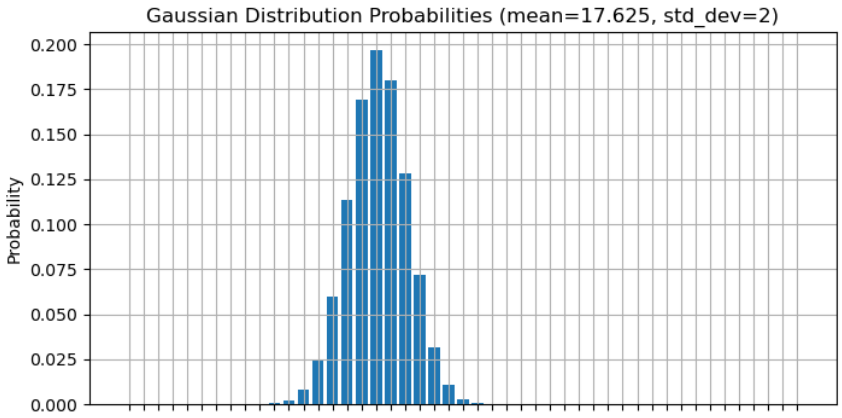


Figure 7: Example of a Gaussian label distribution for a client, centered around a specific label with a fixed standard deviation.

This setup ensures a diverse and realistic distribution of data among clients, which is critical for evaluating the performance of federated learning models in handling non-IID data distributions, as visualized in Figure 7.

Training Settings The experiments are realized using 12 participating clients, training locally for 1 epoch and a learning rate of 0.01 during 100 rounds.

Performance Metric In this experiment, the performance of the global model was evaluated using two key accuracy metrics: *global accuracy* and *individual accuracy*. These metrics were designed to assess how well the model generalizes across all clients and adapts to the specific data distribution of each client.

To calculate these metrics, the accuracy of the global model was first measured on each client's local dataset before any local training was performed. This initial evaluation provided a baseline accuracy, referred to as the *global accuracy*. After the client trained the model on its local data, the accuracy was measured again, reflecting the model's performance after it had been adapted to the client's specific data. This post-training accuracy is termed the *individual accuracy*.

Finally, to obtain an overall measure of model performance across all clients, the server averaged the pre-training (global) and post-training (individual) accuracies separately across all clients. These averaged accuracies provided insight into both the generalizability of the model (through global accuracy) and its ability to adapt to the unique data distributions of individual clients (through individual accuracy).

3.2.3 Experimental Results

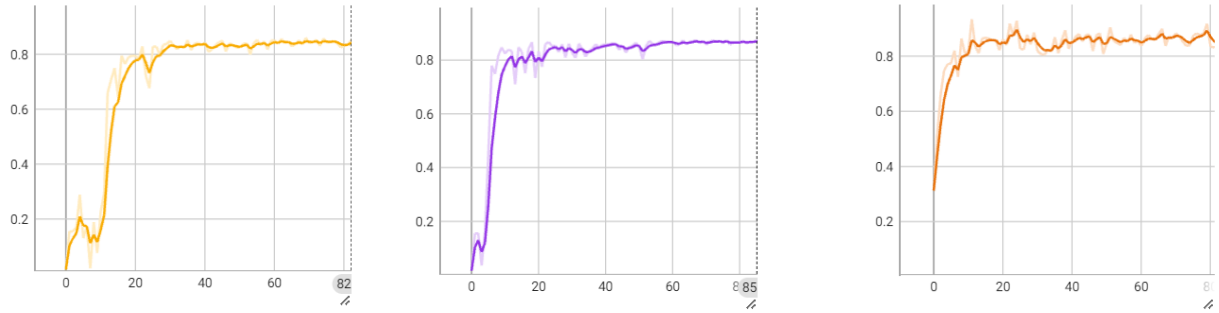
Global model performance in homogeneous settings

The accuracy curves presented in Figure 8 reflect the performance of the global model under homogeneous data distribution conditions, using different training methods.

For the **parallel homogeneous training** method, depicted in Figure 8(a), the accuracy curve demonstrates a rapid rise during the early rounds, stabilizing slightly above 0.8. However, compared to the sequential method, this curve exhibits more fluctuations, particularly in the initial stages, indicating some instability before eventually stabilizing. Although this method achieves a high final accuracy, the increased fluctuations suggest that the model updates may be more sensitive to individual client contributions, leading to less stability during the early training phases.

In the **sequential homogeneous training** scenario, shown in Figure 8(b), the accuracy curve shows a rapid increase in the early stages, with the accuracy quickly stabilizing around 0.8. The model achieves high accuracy after just a few rounds, and the fluctuations in the curve are minimal, indicating stable convergence. This suggests that the sequential method is highly effective when all clients have similar (homogeneous) data distributions, allowing the global model to learn effectively from consistent inputs.

In the case of **fully sequential homogeneous training**, as shown in Figure 8(c), the accuracy curve reaches a high accuracy level, stabilizing just above 0.8, with the least fluctuation among the three methods during the initial rounds. This method shows stable convergence once the model begins to learn effectively. The fully sequential approach, when applied to homogeneous data, demonstrates that it can achieve high accuracy with minimal variance. The stability of this method might be due to the sequential handling of clients, which allows for thorough integration of each client's contribution before moving on to the next. This analysis highlights that homogeneous data distribution across clients allows all methods to perform well, but with varying degrees of efficiency and stability.



(a) Parallel Homogeneous (b) Sequential Homogeneous (c) Fully Sequential Homogeneous

Figure 8: Global accuracy curves for homogeneous data distribution: (a) shows parallel training, (b) shows sequential training, and (c) shows fully sequential training.

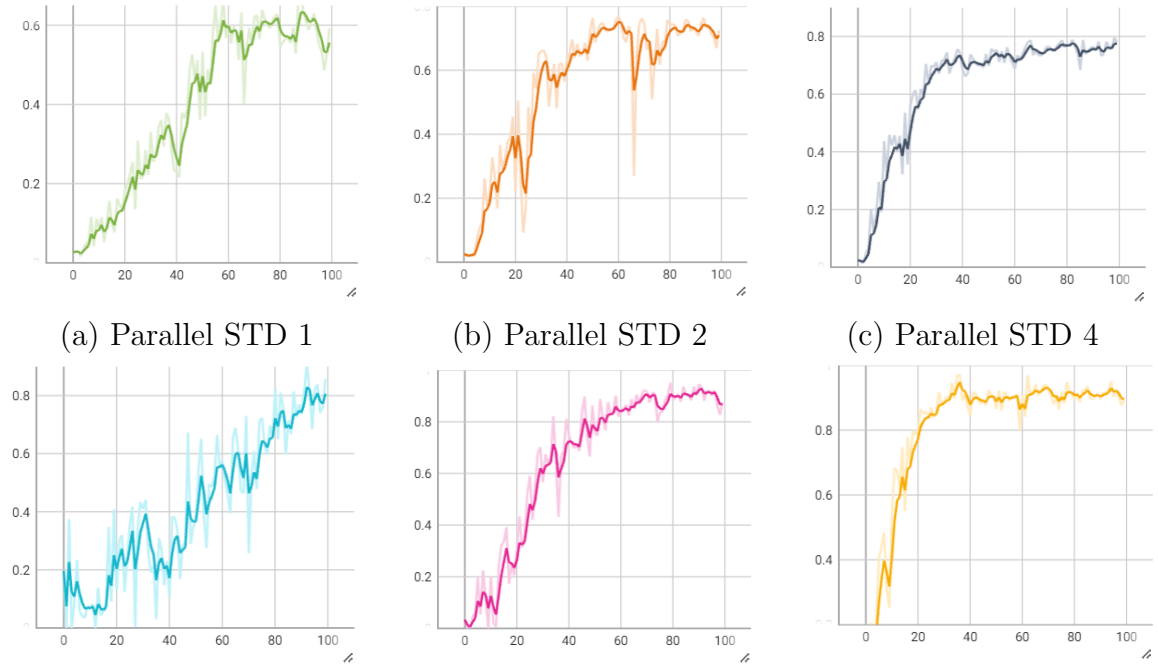
Global model performance in heterogeneous settings

The accuracy curves presented in Figure 9 provide valuable insights into the performance of the global model under different training regimes: parallel training and fully sequential training, each applied with varying standard deviations (STDs) for the label distributions across clients.

In the parallel training scenarios, as depicted in Figures (a) to (c), the global model shows robust performance across all STD settings. In the case of a smaller STD (Figure (a) Parallel STD 1), the accuracy curve begins with significant fluctuations, which is expected due to the narrow distribution of labels. As the standard deviation increases to 2 (Figure (b) Parallel STD 2), the model exhibits fewer fluctuations in the early stages of training and achieves a higher final accuracy of approximately 0.8. This suggests that a broader distribution of labels allows the model to generalize better across different clients. The most pronounced improvement is observed with the largest standard deviation of 4 (Figure (c) Parallel STD 4), where the accuracy increases rapidly and stabilizes around 0.9.

In the fully sequential training scenarios (Figures (d) to (f)), even better results are achieved. With a smaller STD (Figure (d) Fully Sequential STD 1), the final accuracy reaches around 0.8. This better performance indicates that the fully sequential approach may be better suited to narrow label distributions, as the model updates are more heavily influenced by the data from individual clients.

As the STD increases to 2 (Figure (e) Fully Sequential STD 2), the model's performance improves, with the accuracy stabilizing around 0.85. In the scenario with the largest STD of 4 (Figure (f) Fully Sequential STD 4), the accuracy curve demonstrates a more rapid increase with fewer fluctuations, eventually stabilizing around 0.9. As mentioned in [4], sequential training is more efficient than parallel training in extracting useful insights from extremely heterogeneous distributions.



(a) Parallel STD 1 (b) Parallel STD 2 (c) Parallel STD 4

(d) Fully Sequential STD 1 (e) Fully Sequential STD 2 (f) Fully Sequential STD 4

Figure 9: Accuracy curves for different settings: (a-c) show parallel training with standard deviations of 1, 2, and 4, respectively; (d-f) show fully sequential training with the same standard deviations.

Overall, the fully sequential training method consistently outperforms the parallel method in terms of both stability and final accuracy, particularly as the standard deviation of the label distribution increases. The sequential method appears to contribute to a

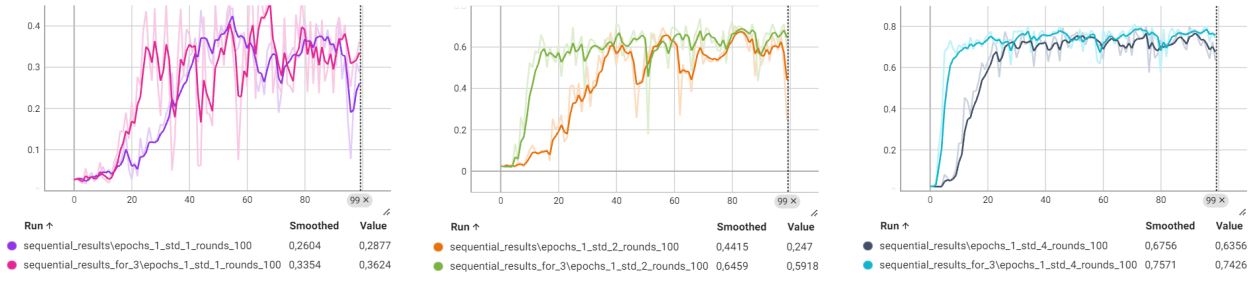
more robust and generalized model. In contrast, the parallel method is more affected by the variability in client data, especially when the label distribution is narrow, resulting in less stable convergence and lower final accuracy.

Comparison with Sequential Training

When comparing the sequential training results presented in Figure 10 with the previously discussed parallel training results, several differences emerge. Sequential training generally exhibits faster convergence and higher variance in the early stages of training compared to parallel training, as seen in the initial fluctuations of the curves. As training progresses, sequential methods tend to stabilize, eventually achieving comparable final accuracies. The slower convergence in the sequential case, in comparison with the fully sequential training, may be attributed to the length of the sequence the model learns from during the training. The longer the sequence, the better the model's performance. In the next paragraph, we further investigate this hypothesis.

Impact of Increasing R

The impact of increasing the number of repetitions R in sequential training is evident in the accuracy curves. As shown in Figure 10, increasing R from 1 to 3 leads to a noticeable improvement in both convergence speed and final accuracy across all standard deviations. This enhancement is particularly pronounced in the higher standard deviation settings (e.g., STD 4), where the model trained with $R = 3$ achieves significantly higher accuracy earlier in the training process. The increased number of repetitions allows the model to refine its updates more effectively, leveraging the diverse data contributions from multiple clients. Consequently, this iterative refinement process helps the model to converge to a more accurate solution, reducing the impact of any single client's data on the final global model.



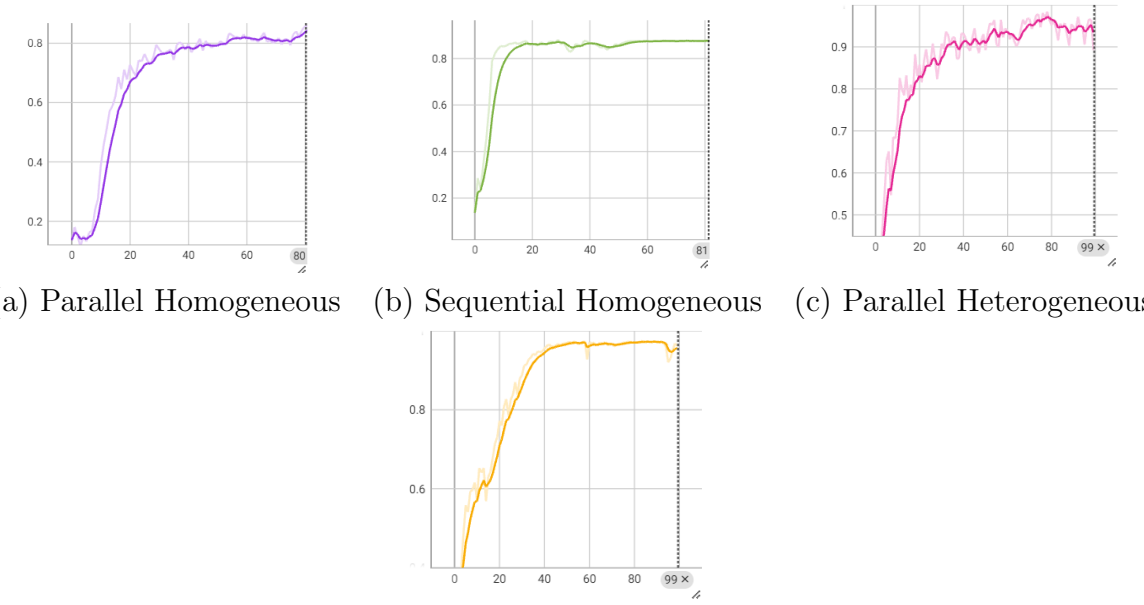
(a) Sequential STD 1 (b) Sequential STD 2 (c) Sequential STD 4

Figure 10: Accuracy curves for sequential training with different values of R: (a) STD 1, (b) STD 2, (c) STD 4. Each plot compares sequential training with R=1 (solid line) and R=3 (dotted line).

Generalization performance

The accuracy curves presented in Figure 11 highlight the differences in performance between homogeneous and heterogeneous data settings, focusing on individual and global accuracy metrics.

In the homogeneous data setting, both individual and global accuracy curves show close alignment, indicating that the performance of individual clients is very similar to the global model's performance. This suggests that in a homogeneous scenario, where clients have similar data distributions, each client's model update contributes effectively to the global model, allowing for strong generalization across all clients. In both parallel and sequential homogeneous settings, the individual accuracy closely follows the global accuracy, demonstrating minimal divergence between the two. The high and consistent performance across clients suggests that the global model is well-representative of the data each client holds. As a result, clients benefit from a shared model that generalizes well. The sequential update process, like the parallel method, does not introduce significant variability, ensuring that clients achieve high accuracy comparable to the global model. This indicates that both methods are effective when the data distribution is uniform across clients.



(a) Parallel Homogeneous (b) Sequential Homogeneous (c) Parallel Heterogeneous

(d) Sequential Heterogeneous

Figure 11: Individual accuracy curves for different settings: (a) Parallel Homogeneous, (b) Sequential Homogeneous, (c) Parallel Heterogeneous, (d) Sequential Heterogeneous.

In the heterogeneous data setting, the gap between individual and global accuracy becomes more pronounced. The individual client performance is more variable, reflecting the challenges posed by non-IID (non-Independent and Identically Distributed) data distributions. In both parallel and sequential heterogeneous settings, there is a noticeable divergence between individual and global accuracy, with the sequential method showing an even larger gap. While the individual accuracy performs well overall, the global model often experiences lower accuracy due to differences in the clients' local data distributions. As a result, clients with data distributions that differ significantly from the global average are less able to generalize, leading to increased variability in performance and lower individual accuracy.

The analysis reveals that in a homogeneous data setting, the alignment between in-

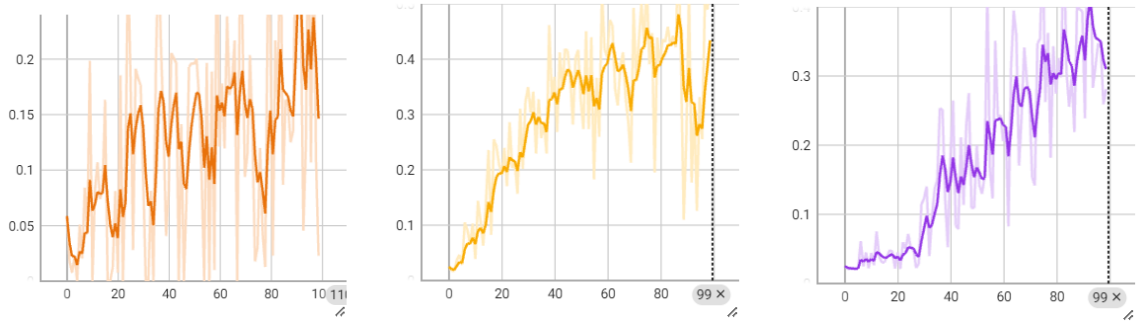
dividual and global accuracy is strong, indicating effective generalization across clients. However, in a heterogeneous setting, data distribution differences prevent some clients from achieving the same level of performance as the global model, leading to a wider gap between individual and global accuracies. This highlights the challenges of federated learning in non-IID settings, where the diversity of client data can hinder the generalization capabilities of the global model.

Impact of byzantine attacks

The accuracy curves in Figure 12 illustrate the impact of a label flipping attack under different training strategies.

In the fully sequential setup (Figure 12a), we observe considerable fluctuations in the accuracy throughout the training process. The sequential nature of the update propagation exacerbates the impact of corrupted labels, as each subsequent client is influenced by the previous model's inaccuracies, leading to a cumulative negative effect. This results in unstable and lower accuracy compared to non-adversarial scenarios.

In the parallel training setting (Figure 12b), the impact of the label flipping attack is somewhat mitigated by the simultaneous updates from all clients. Although the accuracy curve still shows significant variability, the parallel aggregation of updates helps to average out some of the noise introduced by the adversarial clients, leading to slightly more stable performance.



(a) Fully Sequential BLF (b) Parallel BLF (c) Sequential BLF

Figure 12: Accuracy curves under Label Flipping Attack: (a) Fully Sequential Training, (b) Parallel Training, (c) Sequential Training.

The sequential training with $R = 1$ (Figure 12c) shows that the label flipping attack continues to disrupt the learning process. Similar to the fully sequential case, the accuracy curve demonstrates high variability, although it is slightly less severe than in the fully sequential scenario. The main difference here is that the sequential strategy introduces some level of correction between iterations where the algorithm omits some clients from the training process to include only the best-performing clients in the training process, but the overall impact remains detrimental to the final model's accuracy.

The label flipping attack poses a significant challenge across all training methods. However, parallel and sequential training show relatively more resilience due to the concurrent updates, which help dampen the adverse effects of the attack.

4 Conclusion

In this study, we have explored the efficacy of various aggregation methods and training strategies within the context of federated learning, particularly under adversarial conditions involving Byzantine clients. The results indicate that **Centered Clipping (CC)** consistently emerges as the most robust aggregation method, demonstrating resilience against diverse types of Byzantine attacks, including Sign Flipping, Label Flipping, and Gaussian attacks. **Coordinate-wise Trimmed Mean (CwTM)** also showed commendable performance, particularly in scenarios with a higher fraction of Byzantine clients, whereas simpler methods like **Mean Aggregator** and **Comparative Gradient Elimination (CGE)** struggled, highlighting their vulnerability to malicious behavior.

Furthermore, we investigated the performance of sequential training algorithms against parallel training across homogeneous and heterogeneous data distributions. The **Sequential Federated Training** method proved to be superior, especially in handling heterogeneous data, where it outperformed parallel training in terms of both stability and final accuracy. This highlights the potential of sequential training to extract meaningful insights from extremely heterogeneous data distributions.

However, when subjected to adversarial conditions, such as the label flipping attack, all training methods experienced performance degradation, though sequential and parallel training exhibited relatively better resilience due to their ability to mitigate some of the attack's effects through concurrent updates.

The findings underscore the importance of selecting robust aggregation methods and training strategies that are tailored to the specific characteristics of the data distribution and potential adversarial threats in federated learning environments. The **CC** method and **Sequential Training** stand out as particularly effective choices, offering a balance between stability, resilience, and performance in heterogeneous scenarios.

References

[1] B. A. A. B. M. B. A. N. B. K. B. Z. C. G. C. R. C. R. G. D. H. E. S. E. R. D. E. J. G. Z. G. A. G. B. G. P. B. G. M. G. Z. H. C. H. L. H. Z. H. B. H. J. H. M. J. T. J. G. J. M. K. J. K. A. K. F. K. S. K. T. L. Y. L. P. M. M. M. R. N. A. R. P. M. R. H. Q. D. R. R. R. D. S. W. S. S. U. S. Z. S. A. T. S. F. T. P. V. J. W. L. X. Z. X. Q. Y. F. X. Y. H. Y. S. Z. Peter Kairouz, H. Brendan McMahan, "Advances and open problems in federated learning," *IEEE*, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/9464278

[2] T. G. Kunlong Liu, "Federated learning with differential privacy and an untrusted aggregator," *IEEE Transactions on Information Forensics and Security*, 2023. [Online]. Available: https://arxiv.org/abs/2312.10789

[3] L. C. Q. Y. Alysa Ziying Tan, Han Yu, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022. [Online]. Available: https://arxiv.org/abs/2103.00710

[4] D. C. B. C. M. C. Andrea Silvi, Andrea Rizzardi, "Accelerating federated learning via sequential training of grouped heterogeneous clients," *IEEE Access*, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3387453

[5] X. L. Yipeng Li, "Convergence analysis of sequential federated learning on heterogeneous data," *NeurIPS 2023*, 2024. [Online]. Available: https://arxiv.org/abs/2311.03154

[6] H. Chen, Y. Zhang, D. Krompass, J. Gu, and V. Tresp, "Feddat: An approach for foundation model finetuning in multi-modal heterogeneous federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI, 2023. [Online]. Available: https://arxiv.org/abs/2308.12305

[7] Z. L. D. C. D. G. X. P. Y. X. Y. L. B. D. J. Z. Weirui Kuang, Bingchen Qian, "Federatedscope-llm: Federated fine-tuning of large language models under heterogeneous data," *arXiv preprint arXiv:2309.00363*, 2023. [Online]. Available: https://arxiv.org/abs/2309.00363

[8] B. Q. L. Y. Y. L. Jiamu Bai, Daoyuan Chen, "Federated fine-tuning of large language models under heterogeneous data," *arXiv preprint arXiv:2402.11505*, 2024. [Online]. Available: https://arxiv.org/abs/2402.11505

[9] N. G. Rachid Guerraoui and R. Pinot, "Byzantine machine learning: A primer," *ACM*, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3616537

[10] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-Robust federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/fang

[11] M. J. Lie He, Sai Praneeth Karimireddy, "Byzantine-robust decentralized learning," *arXiv preprint arXiv:2202.01545*, 2022. [Online]. Available: https://arxiv.org/abs/2202.01545

[12] M. J. Sai Praneeth Karimireddy, Lie He, "Learning from history for byzantine robust optimization," *arXiv preprint arXiv:2012.10333*, 2021. [Online]. Available: https://arxiv.org/abs/2012.10333

[13] M. Z. M. S. A. T. V. S. Tian Li, Anit Kumar Sahu, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018. [Online]. Available: https://arxiv.org/abs/1812.06127

[14] W. G. D. H. A. I. V. I. C. K. J. K. S. M. H. B. M. T. V. O. D. P. D. R. J. R. Keith Bonawitz, Hubert Eichner, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019. [Online]. Available: https://arxiv.org/abs/1902.01046

[15] M. J. M. J. S. P. K. Mariel Werner, Lie He, "Provably personalized and robust federated learning," *arXiv preprint arXiv:2306.08393*, 2023. [Online]. Available: https://arxiv.org/abs/2306.08393

[16] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2017. [Online]. Available: https://arxiv.org/pdf/1602.05629

[17] R. G. Peva Blanchard, El Mahdi El Mhamdi and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *NIPS 2017*, 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf

[18] Y. F. R. J. F. L. Z. Z. Wang Tian Xiang, Meiyue Shao, "Federated learning framework based on trimmed mean aggregation rules," *SSRN*, 2023. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4181353

[19] F. Lai, Y. Dai, S. Singapuram, J. Liu, X. Zhu, H. Madhyastha, and M. Chowdhury, "Fedscale: Benchmarking model and system performance of federated learning at scale," in *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 2022. [Online]. Available: https://proceedings.mlr.press/v162/lai22a.html