Designing Intelligent Byzantines:
Fall of Robust Aggregators in Federated Learning

by

Ahmet Kerem Özfatura

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Engineering

KOÇ ÜNİVERSİTESİ

1 October, 2024

Designing Intelligent Byzantines:
Fall of Robust Aggregators in Federated Learning

Koç University
Graduate School of Sciences and Engineering
This is to certify that I have examined this copy of a master's thesis by

Ahmet Kerem Özfatura

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____
Assoc. Prof. Alptekin küpçü

_____
Prof. Deniz Gündüz

_____
Asst. Prof. Mehmet Emre Gürsoy

Date: _____

ABSTRACT

Designing Intelligent Byzantines:
Fall of Robust Aggregators in Federated Learning
Ahmet Kerem Özfatura
Master of Science in Computer Engineering
1 October, 2024

In federated learning (FL), it is difficult to profile and verify each client, leading to a security threat, where by malicious clients, called Byzantines, may hamper the accuracy of the trained model by conveying poisoned models during training. Hence, the aggregation process at the parameter server should aim to minimize the detrimental effects of malicious clients. The Byzantine problem is typically analyzed from an outlier detection perspective, and is oblivious to the architecture of the neural network (NN) being trained. In this work, we first expose vulnerabilities of the robust aggregators, specifically the CC framework, and introduce novel attack strategies that can circumvent the defences of the state-of-the-art robust aggregators. Later, we argue that by extracting certain side information specific to the NN architecture, one can design even stronger attacks. Hence, inspired by sparse neural networks, we introduce a hybrid sparse Byzantine attack that is composed of two parts each targeting a different type of defence mechanism: one sparse part that attacks only certain NN parameters with higher sensitivity, and the other being more silent but accumulating over time. Then, we propose a new robust and fast defence mechanism that is effective against the proposed and other existing Byzantine attacks. Our code is publicly available here https://github.com/CRYPTO-KU/FL-Byzantine-Library

ÖZETÇE

Akıllı Bizans Saldırıları Tasarlamak:
Federasyonlu Öğrenmede Dayanıklı Toplayıcıların Çöküşü
Ahmet Kerem Özfatura
Bigisayar Mühendisliği, Yüksek Lisans
1 Ekim 2024

Federasyonlu öğrenmede (FL), her istemciyi profillemek ve doğrulamak zordur ve bu da Bizans saldırganı olarak adlandırılan kötü niyetli istemcilerin eğitim sırasında zehirli modeller ileterek eğitilen modelin doğruluğunu engelleyebileceği bir güvenlik tehdidine yol açar. Bu nedenle, parametre sunucusundaki toplama işlemi kötü niyetli istemcilerin zararlı etkilerini en aza indirmeyi hedeflemelidir. Bizans sorunu genellikle istatistiksel aykırı değer tespiti perspektifinden analiz edilir ve eğitilen sinir ağının mimarisinden habersizdir. Bu çalışmada, öncelikle sağlam toplayıcıların, özellikle CC çerçevesinin güvenlik açıklarını ortaya koyuyoruz ve son teknoloji sağlam toplayıcıların savunmalarını aşabilecek yeni saldırı stratejileri sunuyoruz. Daha sonra, kullanılan sinir ağının mimarisine özgü belirli bir yan bilgi çıkarılarak daha da güçlü saldırılar tasarlanabileceğini savunuyoruz. Bu nedenle, seyrek sinir ağlarından esinlenerek, her biri farklı bir savunma mekanizmasını hedefleyen iki parçadan oluşan bir hibrit seyrek Bizans saldırısı sunuyoruz: yalnızca belirli NN parametrelerine daha yüksek hassasiyetle saldıran seyrek bir parça ve daha sessiz ancak zamanla biriken diğer parça. Ardından, önerilen ve diğer mevcut Bizans saldırılarına karşı etkili olan yeni, sağlam ve hızlı bir savunma mekanizması öneriyoruz. Kodumuz herkese açıktır https://github.com/CRYPTO-KU/FL-Byzantine-Library

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor Dr. Alptekin Küpçü for his guidance, continuous support, belief in me. Dr. Küpçü always supported my ideas and my preferred research direction trough my all master. His support and belief in me allowed me to contribute the science for the topics that I have highly motivated. I feel extremely privileged and honored to have Dr. Alptekin Küpçü as my advisor.

I would like to express my gratitude to Prof. Deniz Gündüz. From starting my internship at Information Processing and Communication Lab to this date, he guided me to become a excellent researcher. His contributions to my research and academic writing is the one the key factor to success of our research.

I am also thankful to my jury member Dr. Emre Gürsoy. His guidance and feedback on my backdoor generation for vision classification task help us achieve great results on the respective field. Additionally, with his leadership on Privacy and security group meetings, I have broaden my understandings on various research methods and subjects, specifically privacy related.

I acknowledge the scholarship provided by KUIS AI Center in the scope of the AI Fellowship Program, and my dear friends in KUIS AI LAB.

Finally, I deeply grateful for my brother Dr. Emre Özfatura, whom supported me as big brother and as a advisor. He contributed greatly to my research and my career itself. His guidance allow me to profound understandings in machine learning and deep learning subjects.

This thesis was supported in part by TÜBİTAK, the Scientific and Technological Research Council of Turkey, under Project 119E088.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

FL	federated learning
PS	Parameter server
NN	Neural network
CC	centered clipping
S-CC	Sequential centered clipping
ROP	Relocated Orthogonal Perturbation
HSA	Hybrid Sparse Attack
HMS	Hybrid MinSum Attack

Chapter 1

INTRODUCTION

Federated learning (FL) [McMahan et al., 2017] has become the de-facto *collaborative learning* framework where a large number of clients iteratively optimize a shared, e.g., neural network (NN) weights, by utilizing their local data and then seek consensus on the global model through the exchange of local models. While local optimization helps keep the personal data private, the consensus phase helps reduce the generalization error. In this work, we focus on the *centralized* and *synchronous* FL setup [McMahan et al., 2017, Konečnỳ et al., 2016], where a central parameter server (PS) orchestrates the consensus phase by executing an aggregation policy for the local models received from the participating clients. The new model is conveyed to the clients for the next round of local optimization/training.

FL is designed to facilitate collaborative learning over a large number of clients, and it is often challenging to profile and verify all the clients. Some clients may act maliciously and aim to sabotage the learning process either individually or by colluding with other malicious clients. In the literature, this is referred to as a poisoning attack [Bagdasaryan et al., 2020, Bhagoji et al., 2019, Fung et al., 2020, Sun et al., 2019], where the malicious clients send a modified model to the PS in compliance with a predefined attack mechanism. These attacks can be *targeted* [Xie et al., 2020a, Bagdasaryan et al., 2020, Bhagoji et al., 2019] with the objective is to make the model fail only certain tasks, e.g., misclassification of certain classes, or *untargeted* [Blanchard et al., 2017, Xie et al., 2020b] so that fails any task [Bhagoji et al., 2019, Fung et al., 2020, Sun et al., 2019, Baruch et al., 2019]. Besides, the attack is not always visible during training; the model may perform well for the training data but lacks generalization at test time, particularly performing poorly on samples with a certain triggering pattern, often referred to as a backdoor at-

tack [Bagdasaryan et al., 2020].

In the scope of this work, we limit our focus to untargeted model poisoning attacks, often known as Byzantine attacks. Design of Byzantine attacks under various assumptions, e.g., regarding the training data that adversaries can access, or the information they can infer, has been extensively studied in the literature [Baruch et al., 2019, Fang et al., 2020, Shejwalkar and Houmansadr, 2021, Xie et al., 2023]. To overcome the detrimental impact of malicious clients, the design of a robust aggregator that neutralizes poisoned models from different aspects has been considered [Blanchard et al., 2017, Pillutla et al., 2022, Yin et al., 2018]. Many of the defence methods treat poisoned models as outliers and suggest eliminating statistically suspicious updates from the clients before using conventional aggregation functions like averaging [El Mhamdi et al., 2018, Blanchard et al., 2017, Muñoz-González et al., 2019]. Some of the prior works redesign the aggregation function to generate a good representation model for benign clients, e.g., geometric mean [Yin et al., 2018, Chen et al., 2020]. Rather than eliminating suspicious models, it is also possible to sanitize all the models before aggregation, as suggested in [Karimireddy et al., 2021, Raynal et al., 2023, El-Mhamdi et al., 2020, Gupta and Vaidya, 2020]. Other more complicated defence strategies either utilize extra data at the PS [Cao et al., 2021a, Xie et al., 2019], or profile the clients over time and assign a trust score to each of them [Muñoz-González et al., 2019, Bhagoji et al., 2019]. Recent works also explore a game-theoretic approach to dynamically select a different robust aggregator at each iteration [Xie et al., 2023].

From the attacker's perspective, an attack should be effective yet imperceptible [Baruch et al., 2019, Shejwalkar and Houmansadr, 2021, Fang et al., 2020]. Hence, attacks often take advantage of the curse of dimensionality due to a large number of being trained, and the variation across benign models due to statistically distinct data distribution across the clients. As the statistical variation among benign models increases, so does the feasible attack space for the Byzantines, as they do not look like outliers. Thus, variance reduction techniques originally designed for stable distributed learning have emerged as effective defence mechanisms by shrinking the feasible attack space for Byzantine clients [Karimireddy et al., 2022, Karimireddy et al., 2021, Gorbunov et al., 2023, Peng et al., 2022b, Wu et al., 2020, Zhu and Ling, 2022, Peng et al., 2022a]. A particularly popular approach is to

employ momentum stochastic gradient descent (SGD) as a local optimizer and to perform aggregation over the momentum terms; it has been shown that the use of local momentum enhances the robustness against Byzantine attacks [Karimireddy et al., 2021, Farhadkhani et al., 2022, El-Mhamdi et al., 2021].

In this work, we first show that the clipping [Karimireddy et al., 2021] mechanism gives a false sense of security, and its performance against ALIE and IMP might be misleading. We show that the existing ALIE and IMP attacks can be redesigned to circumvent the CC defence. We first identify the main vulnerabilities of the CC method, and then, by taking into account certain design aspects of ALIE and IPM, we propose a new Byzantine attack that targets the CC defence. We show numerically that our proposed strategy successfully circumvents the CC defence, as well as other well-known defense mechanisms (i.e., Robust FedAVG (RFA) [Pillutla et al., 2022] and Trimmed-Mean (TM) [Yin et al., 2018]). We then propose an improved defence against the proposed attack and show its effectiveness.

Although there are various successful Byzantine attack strategies, there is still a lack of a universal attack design that is effective against different types of defence mechanisms simultaneously. As we discuss in more detail later, while some attacks are more effective against defences that investigate index-wise outlier values, some others target defence mechanisms that utilize geometric distance as a metric to detect anomalies. In this work, we explore whether it is possible to **design a universally effective yet computationally low-cost Byzantine attack strategy**. Another important aspect often ignored in the design of Byzantine attacks is the architecture of the NN being attacked. Depending on the particular architecture, **certain weights might be more vulnerable against perturbations, and this prior knowledge can be utilized when designing the attack**. To the best of our knowledge, this is the first work that utilizes the NN architecture to design Byzantine attacks.

The contributions of this work can be summarized as follows:

- Taking into account the identified vulnerabilities of CC, we introduce a new Byzantine attack called relocated orthogonal perturbation (ROP), which utilizes the broadcasted momentum term to circumvent the CC defence.

- Then, We introduce another novel Byzantine attack strategy consisting of two parts. The first part is designed to avoid defences that rely on index-wise elimination, while the second one targets geometric distance-based defences. Hence, their combination is effective against a large variety of defence strategies and does not require prior knowledge of the deployed defence mechanism.

- Different from existing Byzantine designs, we utilize side information extracted from the NN architecture through a network pruning framework, which locates the potentially more sensitive weights, and to generate stronger yet imperceptible attacks.

- Through extensive simulations over different NN networks, datasets, and defence mechanisms, we show the effectiveness of our approach by reducing test accuracy up to 60% in the case of independent and identically distributed (IID) simulations and diverging the model training completely in the non-IID simulations.

- Finally, we introduce a new defence mechanism against the proposed Byzantine attacks as well as others, with the same asymptotic complexity as the CC aggregator. We show the robustness of the proposed defence mechanism for both IID and non-IID data distributions.

Chapter 2

PRELIMINARIES AND RELATED WORK

2.1 Notation

We use **bold** case to denote vectors, i.e., $\boldsymbol{v}$, and capital calligraphic letters, e.g., $\mathcal{V}$, to denote sets. When we have an ordered set of vectors $\mathcal{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_i \ldots, \boldsymbol{v}_k\}$, we use subscript index to identify the i^{th} vector in the set, $i \in [k]$, and use double subscript $\mathbf{v}_{i,t}$, particularly when it is changing over time/iteration. For the slicing operation, we use $[\cdot]$, such as $\mathbf{v}[j]$ to denote the j^{th} index of a vector. We use $||\cdot||_p$ to denote the p-norm of a vector, which, without a subscript $||\cdot||$ refers l_2 to norm. We use $<\cdot,\cdot>$ for the inner product between two vectors. In Table 2.1, we list the widely used variables in this paper.

2.2 Federated Learning (FL)

FL allows solving a joint problem among a large number of clients by only sharing their local computation while offering a great generalization opportunity thanks to the client's independent data. Using the consensus model, clients share their computation with respect to their local dataset. This computational result, generally gradients, is then sent to the PS to aggregate and update the consensus model. In this phase, PS generally employs federated averaging as an aggregation method. We illustrate the client computation in Fig. 2.1a, and aggregation then broadcasting in Fig. 2.1b. This collaborative learning paradigm can enable a greater generalization of a common problem without breaching the privacy of the clients.

The formal objective of FL is to solve the following parameterized optimization problem over k clients in a distributed manner

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) = \frac{1}{k}\sum_{i=1}^{k} \underbrace{\mathbb{E}_{\zeta_i\sim\mathcal{D}_i} f(\boldsymbol{\theta}, \zeta_i)}_{:=f_i(\boldsymbol{\theta})}, \tag{2.1}$$

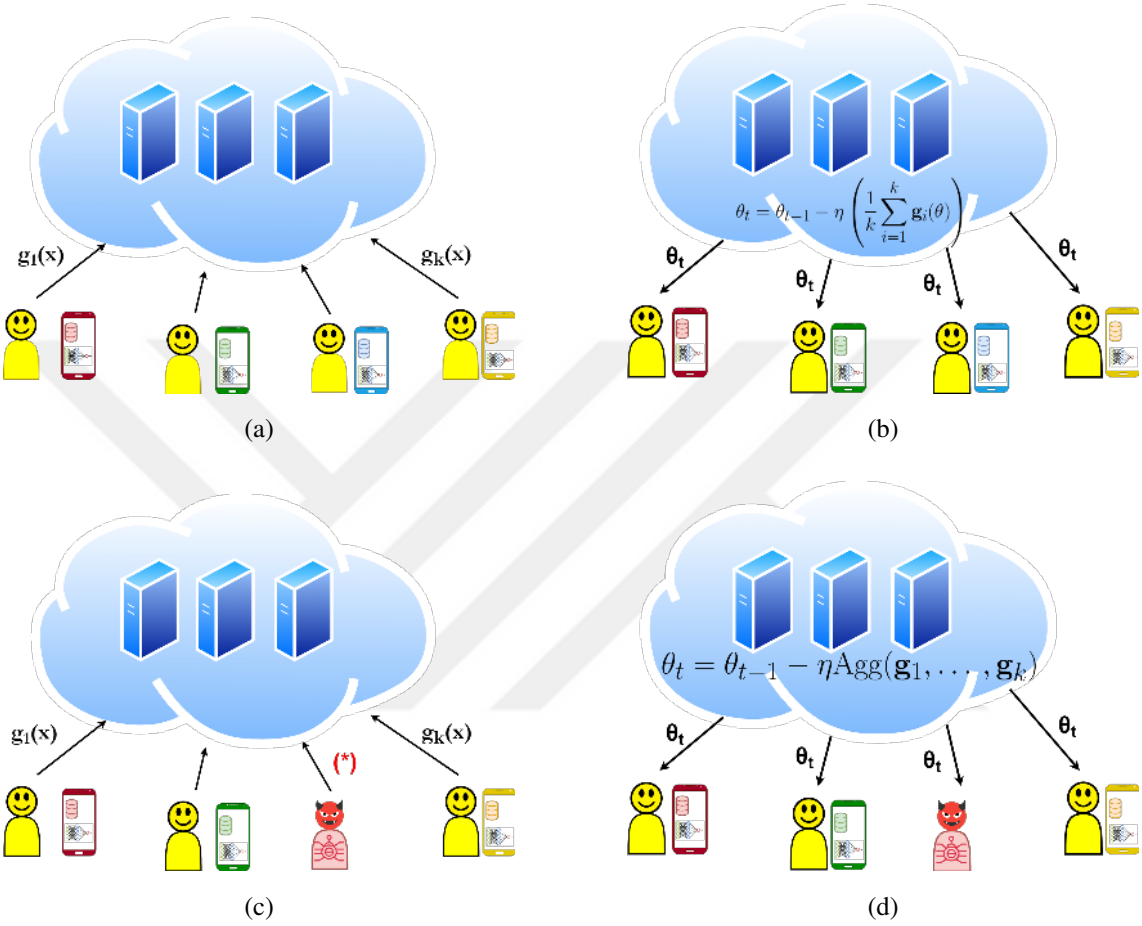


Figure 2.1: Overall Illustration of the federated learning (FL). 2.1a, 2.1b correspond to FL without Byzantine presence where in 2.1a clients send their local updates and in 2.1b clients receive the consensus model from the PS. Without the presence of Byzantines, PS generally employs an algorithm like federated averaging that does not offer Byzantine robustness. 2.1c, 2.1d correspond to FL with the presence of the Byzantines where some of the clients can send malicious updates (2.1c), and PS may employ robust aggregator as a replacement to federated averaging (2.1d) to ensure the robustness of the system before updating the consensus model.

Table 2.1: Notations

Notation	**Description**
$\mathcal{K}_b$, $\mathcal{K}_m$ and $\mathcal{K} = \mathcal{K}_b \bigcup \mathcal{K}_m$	Set of benign, malicious, and all clients, respectively
k_b,k_m and k	Number of benign, malicious, and all clients, respectively
$\boldsymbol{\theta}_{i,t}$, $\mathbf{g}_{i,t}$, and $\mathbf{m}_{i,t}$	Model, gradient, and momentum vector of client i at iteration t, respectively.
$\tilde{\mathbf{m}}_t$ $\hat{\mathbf{m}}_t$ $\bar{\mathbf{m}}_t$,	Reference, aggregated, and benign consensus momentum at iteration t, respectively
δ, $\mathbf{c}$	Sparsity ratio and the corresponding mask vector, respectively

where $\boldsymbol{\theta} \in \mathbb{R}^d$ denotes the model parameters, e.g., weights of a neural network, ζ is a random variable, sample data, from data distribution $\mathcal{D}_i$ of client i, and f is the problem-specific empirical loss function. This is typically solved in an iterative fashion where at each iteration, each client searches for a local model $\boldsymbol{\theta}_i$ that minimizes its local loss function $f_i(\boldsymbol{\theta}_i)$, often by utilizing *SGD* optimizer, then seeks a consensus with other clients to obtain the global model $\boldsymbol{\theta}$ with an assist of the PS.

We note that there are various possible practical implementations of this two-step, local optimization and global consensus, distributed optimization framework, based on chosen local optimization framework e.g., SGD [Polyak, 1964], Adam [Kingma and Ba, 2015], momentum SGD [Sutskever et al., 2013], regularization strategy for local optimization [Ozfatura et al., 2021, Li et al., 2020, Acar et al., 2021], the aggregation strategy at PS, how the local model is communicated with PS, e.g., communicating model [Gao et al., 2022], model difference [Ozfatura et al., 2021], local momentum [Karimireddy et al., 2021, Farhadkhani et al., 2022] and finally based on how the global and local model updates are coordinated [Ozfatura et al., 2021, Gao et al., 2022, Wang et al., 2020, Karimireddy et al., 2020]. Ultimately, all methods ensure the privacy of the clients by not sharing their local data while solving a common optimization problem. However, a honest-but-curious server can still breach the privacy of the clients by launching gradient inversion attacks [Geiping et al., 2020].

In the scope of this work, based on the discussion above, we have focused on a particular setup where each client utilizes momentum SGD for local optimization, i.e, each

client first computes the gradient for a randomly sampled minibatch ζ_i

$$\mathbf{g}_i \leftarrow \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta}, \zeta_i), \tag{2.2}$$

where $\boldsymbol{\theta}$ is the last consensus model, the local gradient $\mathbf{g}_i$ is used for updating the momentum term $\mathbf{m}_i$,

$$\mathbf{m}_i = (1-\beta)\mathbf{g}_i + \beta\mathbf{m}_i, \tag{2.3}$$

where β is a hyper-parameter for the momentum term. Once the local momentum is computed, each client sends their local momentum term to PS for the consensus. Although the presented materials in this work can be extended to different setups, there are two key reasons for focusing on this framework first, the use of momentum helps accelerate gradient vectors in the right directions, thus leading to faster converging [Sutskever et al., 2013, Nesterov, 1983, Polyak, 1964], second and more importantly, the aggregation over local momentum terms for the global model update also offers a certain robustness against Byzantines, since momentum helps for variance reduction, thus it is a more challenging setup for Byzantines [Karimireddy et al., 2021, Farhadkhani et al., 2022, El-Mhamdi et al., 2021, Karimireddy et al., 2022, He et al., 2022]. Next, we briefly explain the adversary model and illustrate the overall framework.

Adversary Model: We assume the adversary controls up to k_m out of k total clients, called malicious clients. We assume that the number of malicious clients is less than the number of benign clients, i.e., $(k_m/k) < 0.5$; otherwise, no Byzantine-robust aggregator will be able to defeat poisoning attacks. Following the previous works [Shejwalkar and Houmansadr, 2021, Bagdasaryan et al., 2020, Baruch et al., 2019, Bhagoji et al., 2019, Fang et al., 2020, Xie et al., 2020b], we assume that the adversary can access the global model parameters broadcast in each epoch and can directly manipulate the gradients on malicious devices. We further assume that Byzantines can also compute the momentum of the benign clients. On the other hand, Byzantines form their attacks in an agnostic manner with respect to the aggregator. In other words, the Byzantines do not know the robust aggregator or the defence mechanism used by the PS. Hence, the attack should be universal; that is, it should be effective against all known defence mechanisms without any aggregator-specific calibration.

Algorithm 1 FL with Robust Aggregator and Byzantines

Input: η, $AGG(\cdot)$, $Attack(\cdot)$
Output: Consensus model: $\boldsymbol{\theta}_T$
for $t = 1, \ldots, T$ **do**
    Client side:
    for $i = 1, \ldots, k$ **do** in parallel
        Receive: $\boldsymbol{\theta}_{t-1}$ from PS
        if $i \in \mathcal{K}_b$ **then**
            Compute gradient estimate: $\mathbf{g}_{i,t} \leftarrow \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta}_{t-1}, \zeta_{i,t})$
            Update Momentum: $\mathbf{m}_{i,t} = (1-\beta)\mathbf{g}_{i,t} + \beta \mathbf{m}_{i,t-1}$
        else
            $\mathbf{m}_{i,t} \leftarrow Attack(\mathcal{H}_t)$
    Server side:
    Aggregate local updates: $\tilde{\mathbf{m}}_t \leftarrow AGG(\mathbf{m}_{1,t}, \ldots, \mathbf{m}_{k,t})$
    Update the model: $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \tilde{\mathbf{m}}_t$

In this setup, the flow of the overall framework with Byzantines performing an attacking strategy, denoted by $Attack(\cdot)$, and the robust aggregator, denoted by $AGG(\cdot)$ that is implemented at the server In general, the attack $Attack(\cdot)$ can be considered as a mapping

$$Attack(\cdot) : \mathcal{H}_t \rightarrow (\mathbf{m}_{i,t}\ i \in \mathcal{K}_{mal}) \ \in \mathbb{R}^d, \tag{2.4}$$

where $\mathcal{H}_t$ is referred to as the history and includes all the available information to client i until iteration t. The overall setup is illustrated in Algorithm 1. FL with the presence of the Byzantines is also illustrated in Fig. 2.1c and 2.1d

2.3 Existing Defence Methods

Various defence mechanisms against Byzantines have been introduced to ensure the learned model's reliability. In a broad sense these frameworks can be analyzed under two category, namely elimination and sanitization. In the first category, defence mechanisms are designed to detect Byzantines updates, by assuming these updates exhibit outlier behavior

Table 2.2: Comparison of the defence methods

Defence mechanism	**Identification Metric**	**Defence**
Centered Clipping [Karimireddy et al., 2021]	Euclidean distance	Sanitation (clipping)
Krum & Multi-Krum [Blanchard et al., 2017]	Euclidean distance	Elimination
RFA [Pillutla et al., 2022]	Euclidean distance	Sanitation (Geometric median)
Bulyan [El Mhamdi et al., 2018]	Euclidean distance	Elimination
Trimmed mean [Yin et al., 2018]	Index-wise statistics	Elimination
Centered median [Yin et al., 2018]	Index-wise statistics	Sanitaion (Index-wise Median)
SignSGD [Bernstein et al., 2019, Jin et al., 2024]	Index-wise statistics	Sanitation (Majority voting)
FL-trust [Cao et al., 2021b]	Angular inspection	Sanitation and Elimination

compared to benign updates, by utilizing different metrics for outlier detection such as geometric distance, index-wise statistics, and angular variance. Once the outlier values are eliminated, over the vectors or indices, the remaining values are aggregated. These methods often assume the number of Byzantines is known by the server, and the key drawback of such defences is the elimination of the benign updates due to miss-classification. Alternatively, defence strategies in the second category utilize all updates but sanitize them to neutralize Byzantines' detrimental impact through different mechanism such as majority voting, geometric median.

For the methods following the former approach, Byzantine detection strategies can be grouped into three main categories based on the identification measures used for outliers: geometric distance, index-wise statistics, and angular variance. We briefly identify the defence mechanisms that we utilized in the Table 2.2.

2.3.1 Geometric distance

We first present the norm-based defences where robust aggregators consider the geometric distances to identify outliers.

Multi Krum: The underlying idea behind the Krum mechanism [Blanchard et al., 2017] is to identify where the benign vectors accumulate in close proximity. Under the assumption of k_m Byzantines, considered as outliers, the first step is to form a set of neighboring clients $\mathcal{S}_i$, for each vector $\mathbf{m}_i$, with cardinality $k^\star = k - k_m + 2$, such that

$$\mathcal{S}_i^\star = \underset{\mathcal{S},|\mathcal{S}|=k^\star}{\operatorname{argmin}} \sum_{j\in\mathcal{S}} \left\|\mathbf{m}_i - \mathbf{m}_j\right\|^2 . \tag{2.5}$$

Then, Krum identifies the vector $\mathbf{m}_i$ that has the minimum average distance to the vectors in its closest neighborhood $\mathcal{S}_i^\star$ as the true center for the benign vectors, i.e.,

$$\operatorname{argmin}_{i\in\mathcal{K}} \sum_{j\in\mathcal{S}_i^\star} \left\|\mathbf{m}_i - \mathbf{m}_j\right\|^2 \tag{2.6}$$

Multi-krum is built on the same principle but chooses n vectors instead of a single one, and outputs their average.

Centered Clipping (CC): One of the major shortcomings of the robust aggregators that rely on outlier elimination is the assumption that the number of Byzantines is known in advance. Even under this assumption, such methods suffer from the miss-classification of benign clients as outliers, or the under-utilization of benign models [Karimireddy et al., 2022, Baruch et al., 2019]. Hence, in [Karimireddy et al., 2021], the authors proposed to utilize all the available model updates after first sanitizing them. The underlying idea behind CC is to perform sanitization by projecting model update $\mathbf{m}$ into a ball with radius τ and centered at reference model $\tilde{\mathbf{m}}$, accordingly, the clipping function can be formulated as

$$f_{CC}(\mathbf{m}|\tilde{\mathbf{m}},\tau) = \tilde{\mathbf{m}} + \min\left\{1, \frac{\tau}{||\tilde{\mathbf{m}} - \mathbf{m}||}\right\}(\mathbf{m} - \tilde{\mathbf{m}}). \tag{2.7}$$

Further, by choosing the previous aggregate model update as the reference model for the current step, i.e., $\tilde{\mathbf{m}}_t = \hat{\mathbf{m}}_{t-1}$, it is possible to sanitize poisoned model updates successfully. However, in 3, we will show that when Byzantines can also modify their adversarial perturbation according to $\tilde{\mathbf{m}}_t$, CC which makes it ineffective. As we further discuss in Section 2.4 and Section 3, CC has other weaknesses as well.

Robust-Federated-Averaging (RFA): is a geometric median-based robust aggregation method [Pillutla et al., 2022], where the consensus model $\mathbf{m}^\star$ is obtained as the solution of the following optimization problem:

$$\text{RFA}(\mathbf{m}_1, \ldots, \mathbf{m}_n) = \underset{\mathbf{m}^\star}{argmin} \sum_{i=1}^{n} ||\mathbf{m}^\star - \mathbf{m}_i||_2. \tag{2.8}$$

It reduces the impact of contaminated updates by using the geometric median in place of the weighted arithmetic mean.

Algorithm 2 Robust aggregation with CC

1: **Inputs:** $\tilde{\mathbf{m}}_t$, $\left\{\mathbf{m}_{i,t}\right\}_{i\in\mathcal{K}}$, τ

2: **for** $i = 1, \ldots, k$ **do** in parallel

3: $\quad \tilde{\mathbf{m}}_{i,t} = f_{CC}(\mathbf{m}_{i,t}|\tilde{\mathbf{m}}_t, \tau)$

4: $\hat{\mathbf{m}}_t = \frac{1}{k}\sum_{i\in\mathcal{K}} \tilde{\mathbf{m}}_{i,t}$

2.3.2 Index-wise statistics

SignSGD. [Bernstein et al., 2019, Jin et al., 2024] Reduces the communication overheads in FL by updating the parameters according to a majority vote on the sign of the momentum terms. It also allows fault tolerance by employing majority voting. More formally, the aggregation rule of SignSGD can be defined as:

$$\text{SignSGD}(\mathbf{m}_1, \ldots, \mathbf{m}_k) = \text{sign}\left(\sum_{i}^{k} \text{sign}(\mathbf{m}_i)\right). \tag{2.9}$$

Coordinate-wise median (CM) [Yin et al., 2018] employs the index-wise median of momentum vectors.

$$\hat{\mathbf{m}} = \text{CM}(\mathbf{m}_1, \ldots, \mathbf{m}_k) : \hat{\mathbf{m}}[j] = \text{median}(\mathbf{m}_1[j], \ldots, \mathbf{m}_k[j]). \tag{2.10}$$

Trimmed mean (TM) [Yin et al., 2018], is proposed as a more robust Byzantine resilient aggregator to CM by discarding some of the outlier indices and then taking the mean of the remaining ones. It offers less statistical error under the assumption that the number k_m of Byzantines is known. For each coordinate, j, let $\mathcal{S}_j$ denote the ordered set formed by permuting the elements in set $[n]$, such that the order is determined by sorting the j coordinate values. We compute the average after discarding the k_m largest and smallest values; hence the name trimming [Yin et al., 2018]:

$$[\text{TM}(\mathbf{m}_1, \ldots, \mathbf{m}_n)]_j = \frac{1}{k - 2k_m} \sum_{i=k_{m+1}}^{k-k_{m-1}} [\mathbf{m}_{\mathcal{S}_\mathbf{j}(\mathbf{i})}]_j \tag{2.11}$$

Based on the variance and skewness of the various workers, the authors of [Yin et al., 2018] bound the error rate TM ensures. As such, the larger the variance and skewness,

the higher the error rate the attacker can achieve. Nevertheless, regardless of the variance, Byzantine clients can steadily accumulate errors as long as staying close to the $\bar{\mathbf{m}}$ [Fang et al., 2020, Shejwalkar and Houmansadr, 2021, Baruch et al., 2019].

Bulyan [El Mhamdi et al., 2018] is designed as a two-staged defence mechanism, wherein the first stage $k - 2k_m$ benign candidates are selected according to an elimination based aggregation rule, often the common approach is Krum, then in the second stage these candidates are aggregated according to TM.

GAS [Liu et al., 2023] highlights that the existing defence mechanisms suffer from the curse of dimensionality and argues that it is possible to enhance the robustness of existing aggregators by dividing the whole update vector into sub-vectors and performing aggregation for each sub-vector independently. The number of the sub-vectors is chosen according to the size of the underlying NN architecture, and often larger values are employed for better robustness.

2.3.3 Angular inspections

FL-Trust [Cao et al., 2021b] updates the global model using the root data, then clips and normalizes the local model updates using the momentum calculated with the root data as a baseline. Each client's momentum is then scaled based on the cosine similarity of the server's momentum before the aggregation. This method increases the aggregated model's reliability and diminishes the impact of adversaries.

2.4 Existing Attack Methods

ALIE: Traditional aggregators such as Krum [Blanchard et al., 2017], TM [Yin et al., 2018], and Bulyan [El Mhamdi et al., 2018] assume that the selected set of momentums will lie within a ball centered at the real mean, whose radius is a function of the number of benign clients. The attack proposed in [Baruch et al., 2019] called ALIE utilizes index-wise mean ($\bar{\mathbf{m}}$) and standard deviation ($\bar{\sigma}$) vectors of the benign clients to induce small but consistent perturbations to the parameters. By keeping the momentum values close to $\bar{\mathbf{m}}$, ALIE can steadily achieve an accumulation of error while concealing itself as a benign client during training. To avoid detection and stay close to the center of the ball,

ALIE scales $\bar{\boldsymbol{\sigma}}$ with a parameter that depends on the number of benign and Byzantine clients. As such, let s be the minimal number $s = \left[\frac{k}{2} + 1\right] - k_m$ of benign clients that are required as *supporters*. The attacker then uses the properties of the normal distribution, specifically the cumulative standard normal function $\phi(z)$, and looks for the maximum value of z, denoted by z^{max}, such that s benign clients will have a greater distance to the mean compared to the Byzantine clients. As a result, the benign clients are more likely to be classified as Byzantines. At a high level, z^{max} can be calculated as:

$$z^{max} = \max \left\{ z : \phi(z) < \frac{k - k_m - s}{k - k_m} \right\} \tag{2.12}$$

Ultimately, z^{max} is employed as a scaling parameter for the standard deviation to perturb the mean of the benign clients in set $\mathcal{K}_b$:

$$\mathbf{m}_i = \bar{\mathbf{m}} - z^{max} \bar{\boldsymbol{\sigma}} \, , \forall i \in \mathcal{K}_m, \tag{2.13}$$

Each Byzantine client generates an attack with a momentum value near $\bar{\mathbf{m}}$, following (2.13).

Inner Product Manipulation (IPM): The authors of [Xie et al., 2020b] highlight that the required condition for the convergence is that the inner product between the benign gradient $\bar{\mathbf{g}}$ and the output of the robust estimator should be positively aligned, i.e.,

$$\langle \bar{\mathbf{g}}, AGG(\mathbf{g}_i : i \, \epsilon \, \mathcal{K}) \rangle \geq 0, \tag{2.14}$$

which ensures that the loss is steadily minimized over iterations. Hence, IPM generates poisoned model updates at each iteration with an objective that the condition in (2.14) is not met. To this end, IPM chooses the benign gradient with the inverse sign, $-\bar{\mathbf{g}}$, as its base for the attack and chooses a proper scaling parameter z for imperceptibility so that the final version of the attack $-z\bar{\mathbf{g}}$ is not easily spotted but hinders the convergence[1]. One of the major drawbacks of IPM is requires a large ratio of $\frac{k_m}{k_b}$ to prevent eqn 2.14.

Adaptive Optimized Attacks: ALIE and IPM attacks utilize a fixed pre-determined scaling parameter throughout the training process. In [Shejwalkar and Houmansadr,

[1]When the aggregation is performed over momentum terms the attack is performed as $-z\bar{\mathbf{m}}$.

2021], the authors utilize ALIE and IPM framework to obtain the base form of the perturbation. However, instead of using a fixed scaling parameter z, they adaptively change it by solving an optimization problem with the objective of maximizing z under certain geometric distance based imperceptibility constraints at each iteration.

Chapter 3

VULNERABILITIES OF CC AND DESIGNING STRONG BUT IMPERCEPTIBLE ATTACK

In this section, we identify the main limitations of the CC defence mechanism and underline certain aspects to design strong but imperceptible attacks.

3.1 Relocation of the attack

Existing Byzantine attacks set $\bar{\mathbf{m}}_t = \frac{1}{k_b} \sum_{i \in \mathcal{K}_b} \boldsymbol{m}_i$ as a reference to design the attack. Hence, by clipping each individual update according to the previous update direction $\tilde{\mathbf{m}}_{t-1}$, it is possible to reduce the impact of a poisoning attack. Let $\boldsymbol{\Delta}_{i,t}$, $i \in \mathcal{K}_m$, denote the attack vector of Byzantine client i at time t to the mean benign update $\bar{\mathbf{m}}_t$.

In the CC framework, the benign local momentum and global momentum evolve as follows:

$$\mathbf{m}_{i,t} = \beta \mathbf{m}_{i,t-1} + (1-\beta)\mathbf{g}_{i,t}, \quad i \in \mathcal{K}_b, \tag{3.1}$$

$$\bar{\mathbf{m}}_t = \sum_{i \in \mathcal{K}_b} \left(\beta \mathbf{m}_{i,t-1} + (1-\beta)\mathbf{g}_{i,t}\right) = \beta \bar{\mathbf{m}}_{t-1} + (1-\beta) \sum_{i \in \mathcal{K}_b} \mathbf{g}_{i,t}. \tag{3.2}$$

Further, let $\tilde{\boldsymbol{\Delta}}_t$ denote the distance between the reference point at $t-1$, $\tilde{\mathbf{m}}_{t-1}$, and the benign momentum at time t, $\bar{\mathbf{m}}_t$, i.e.,

$$\tilde{\boldsymbol{\Delta}}_t = \bar{\mathbf{m}}_t - \tilde{\mathbf{m}}_{t-1}. \tag{3.3}$$

Existing attacks often target the benign momentum $\bar{\mathbf{m}}_t$; hence, the poisoned updates of the Byzantine client can be written in the following form, with respect to $\bar{\mathbf{m}}_t$ and $\tilde{\mathbf{m}}_{t-1}$:

$$\mathbf{m}_{i,t} = \bar{\mathbf{m}}_t + \boldsymbol{\Delta}_{i,t}\ , \quad i \in \mathcal{K}_m, \quad = \tilde{\mathbf{m}}_{t-1} + \underbrace{\tilde{\boldsymbol{\Delta}}_t + \boldsymbol{\Delta}_{i,t}}_{\tilde{\boldsymbol{\Delta}}_{i,t}}, \tag{3.4}$$

where $\tilde{\mathbf{\Delta}}_{i,t}$ is the aggregate form of the attack with respect to the reference point of the CC, $\tilde{\mathbf{m}}_{t-1}$. When $||\tilde{\mathbf{\Delta}}_{i,t}||$ is larger then the radius τ of CC, the aggregation mechanism of CC scales $\tilde{\mathbf{\Delta}}_{i,t}$ with $\frac{\tau}{||\tilde{\mathbf{\Delta}}_{i,t}||}$. The scaled version of the attack can be written as a sum of two components: one towards $\bar{\mathbf{m}}_t$ and the second in the direction of the attack, i.e.,

$$\tilde{\mathbf{\Delta}}_t \frac{\tau}{||\tilde{\mathbf{\Delta}}_{i,t}||} + \mathbf{\Delta}_{i,t} \frac{\tau}{||\tilde{\mathbf{\Delta}}_{i,t}||}. \tag{3.5}$$

When the CC defense is employed, the clipped poisoned update often includes both a benign update and the attack with the corresponding scaling factor in (3.5). We emphasize that the strength of the attack is directly related to the impact of $\mathbf{\Delta}_{i,t}$ in $\tilde{\mathbf{\Delta}}_{i,t}$, which eventually determines the effective scaling of the pure attack $\mathbf{\Delta}_{i,t}$. Significantly reducing $\tilde{\mathbf{\Delta}}_t$ gives the attacker more room to further scale up the perturbation $\mathbf{\Delta}_{i,t}$ until $\frac{\tau}{||\tilde{\mathbf{\Delta}}_{i,t}||} = 1$, while still avoiding clipping. Setting $\mathbf{\Delta}_{i,t} >> \tilde{\mathbf{\Delta}}_t$ maximizes the effectiveness of $\mathbf{\Delta}_{i,t}$ as it can perturb and poison the PS model by staying close to the center of clipping.

Hence, CC around the previous update direction helps suppress attacks targeting the current benign update direction. However, this strong aspect of the CC framework also becomes its vulnerability: if the attacker knows the reference point of CC, it only requires the knowledge of the learning rate, which can be easily predicted, to modify the attack accordingly. In other words, the attack can be generated with respect to the reference point, $\tilde{\mathbf{m}}_{t-1}$, and easily escape clipping.

We refer to this observation as the *target reference mismatch*, since the target of the attacker, which is often the benign update, is different from the reference of the defence mechanism. We further argue that CC relies on this mismatch and induces a false sense of security. Later, we numerically show how CC can be easily fooled if the attack is revised accordingly. To this end, we consider the *relocation of the attack*, simply targeting the previous update instead of the current benign update. By doing so, we show that the accuracy under the CC defence significantly drops. We will further show that such a strategy is not only successful against CC but also against other SotA defense mechanisms such as TM [Yin et al., 2018] and RFA [Pillutla et al., 2022].

3.2 Angular Invariance

One of the major drawbacks of CC is the angular invariance against attacks that target the reference point $\tilde{\mathbf{m}}_t$. The CC performs a scaling operation by clipping the client's momentum $\mathbf{m}_{i,t}$ that lies beyond a certain radius. This is achieved by scaling the gap $\mathbf{\Delta} = \mathbf{m}_{i,t} - \tilde{\mathbf{m}}_{t-1}$ with a factor δ, which depends only on its norm $||\mathbf{\Delta}||$, and it operates as an identity function if $\frac{\tau}{||\mathbf{\Delta}||} \geq 1$.

Now, consider two vectors $\mathbf{m}_1 = \tilde{\mathbf{m}}_{t-1} + \mathbf{\Delta}_1$ and $\mathbf{m}_2 = \tilde{\mathbf{m}}_{t-1} + \mathbf{\Delta}_2$, where $||\mathbf{\Delta}_1|| = ||\mathbf{\Delta}_2|| \leq \tau$, but $\mathbf{\Delta}_1 \neq \mathbf{\Delta}_2$. f_{CC} treats $\mathbf{m}_1$ and $\mathbf{m}_2$ in an identical manner; however, their angle with respect to the reference point, $\tilde{\mathbf{m}}_{t-1}$, can be significantly different.

A fundamental question for a fixed norm constraint, $||\mathbf{\Delta}|| \leq \tau$, is how to decide on the attack $\mathbf{\Delta}$ that has the highest impact? A similar problem is discussed in [Xie et al., 2020b], and from the theoretical analysis of the convergence behavior, the authors argue that for a given benign momentum $\bar{\mathbf{m}}$, the aggregated momentum $\tilde{\mathbf{m}}$ is successfully poisoned if

$$\langle \bar{\mathbf{m}}, \tilde{\mathbf{m}} \rangle < 0, \tag{3.6}$$

in which case the aggregation framework does not meet the required convergence condition. Accordingly, in [Xie et al., 2020b], the authors propose to use a scaled version of the benign gradient $-\epsilon\bar{\mathbf{m}}$, $0 < \epsilon < 1$, as a poisoned gradient. However, as highlighted in [Karimireddy et al., 2021], unless there is a sufficient number of Byzantine clients, it is often difficult to ensure (3.6). To formally illustrate, considering the naive averaging strategy as an aggregator, we have

$$\tilde{\mathbf{m}}_t = \frac{1}{k} \sum_{i \in \mathcal{K}} \mathbf{m}_{i,t} = \frac{1}{k} \left(\sum_{i \in \mathcal{K}_b} \mathbf{m}_i + \underbrace{\sum_{i \in \mathcal{K}_m} \mathbf{m}_i}_{-k_m \delta \bar{\mathbf{m}}_t} \right), \tag{3.7}$$

and we have

$$\mathbb{E}\left[\tilde{\mathbf{m}}_t\right] = \frac{1}{k}\left(k - k_m(1+\delta)\right)\bar{\mathbf{m}} \qquad = \left(1 - \frac{k_m}{k}(1+\delta)\right)\bar{\mathbf{m}}_t. \tag{3.8}$$

Hence, if $\frac{k}{k_m} < (1+\delta)$, then the expected $\tilde{\mathbf{m}}_t$ will be positively aligned with $\bar{\mathbf{m}}$. Nevertheless, as shown in [Xie et al., 2020b] (see Theorem 2), when there is certain

variation among $\mathbf{m}_{i,t}, i \in \mathcal{K}_b$, then IPM can be successful in negatively aligning $\tilde{\mathbf{m}}$ with $\bar{\mathbf{m}}$. Consequently, under certain conditions on the variation among the true gradients, the IPM attack can be successful. On the other hand, when $k >> k_m$, on average $\tilde{\mathbf{m}}_t$ is a scaled version of $\bar{\mathbf{m}}_t$ with a positive coefficient. To conclude, the impact of the IPM attack is highly dependent on the ratio of malicious clients and how the benign clients' gradients/updates are aligned with the benign update direction, i.e., benign clients with very heterogeneous data. Another drawback of the IPM attack is that, although a scaling parameter is used to hide malicious updates, a defence mechanism that utilizes the angular distance rather than a norm-based distance can easily detect malicious clients.

3.3 Importance of temporal correlation

At this point, we revisit another well-known attack strategy called *ALIE* to highlight the key notions behind our attack strategy. Contrary to IPM, ALIE does not specify a direction for the attack with respect to the benign update $\bar{\mathbf{m}}_t$, but introduces a radius τ_i, for each index i, so that the poisoned update can not be arbitrarily far away from the benign one, index-wise, i.e.,

$$|\bar{\mathbf{m}}_t[i] - \mathbf{m}_t^{attack}[i]| \leq \tau_i, \tag{3.9}$$

where τ_i is determined based on the statistics of the ith index of the updates of benign clients as well as the number of Byzantines, which is further discussed in Section 2.4. Contrary to (3.8), on average we have

$$\mathbb{E}\left[\tilde{\mathbf{m}}_t\right] = \bar{\mathbf{m}}_t + k_m \mathbf{\Delta}_t, \tag{3.10}$$

where $\mathbf{\Delta}_t$ is not aligned with $\bar{\mathbf{m}}_t$, indeed, it is often orthogonal to $\bar{\mathbf{m}}_t$. Overall, the objective is to perturb the update direction as much as possible without being detected as an outlier. To achieve consistent error accumulation, the attacker must employ similarly aligned perturbation vectors during training. Here we note that, to measure the alignment of two vectors a common metric is the *cosine similarity*, defined as:

$$\cos(\mathbf{m}_1, \mathbf{m}_2) = \frac{\langle \mathbf{m}_1, \mathbf{m}_2 \rangle}{||\mathbf{m}_1||_2 ||\mathbf{m}_2||_2}. \tag{3.11}$$

Table 3.1: Cosine similarity for the perturbation values of the ALIE attack on IID CIFAR-10 dataset. The average cosine similarity is reported over 100 epochs (6250 communication rounds).

	$\beta = 0$	$\beta = 0.9$	$\beta = 0.99$
$cos(\mathbf{\Delta}_{t-1}, \mathbf{\Delta}_t)$	0.94	0.995	0.999

ALIE is quite effective against TM [Yin et al., 2018], Krum [Blanchard et al., 2017], and Bulyan [El Mhamdi et al., 2018] defense mechanisms [Baruch et al., 2019]. Apart from being statistically less visible, ALIE mostly gains from accumulating the perturbations over time. We argue that the enabling factor behind the accumulation is the correlation of consecutive attacks; in other words, attack vectors are positively aligned over time, i.e., $\cos(\mathbf{\Delta}_t, \mathbf{\Delta}_{t-1}) \approx 1$. We emphasize that, even though the attack is formed independently from the benign gradients without specifying a particular direction; since it directly utilizes the statistics of the benign client updates which often vary slowly during training, the adversarial perturbations end up being highly correlated over time.

We argue that the variance $\bar{\boldsymbol{\sigma}}_t$ from benign clients $\mathcal{K}_b$, thus the direction of the attack vector, does not vary significantly during training. In Table 3.1, we demonstrate that $\mathbf{\Delta}_t$ of ALIE is always aligned positively, which verifies our initial argument that the perturbation $z^{max}\bar{\boldsymbol{\sigma}}_t$ used by ALIE attack is consistent over iterations in terms of its direction. This temporal correlation leads to a stronger accumulation and enhances the strength of the attack. Now, to better visualize the impact of temporal accumulation, we consider a scenario where the sign of $\mathbf{\Delta}_t$ is alternated over consecutive communication rounds;

$$\mathbf{\Delta}_t = \left\{ \begin{array}{ll} z^{max}\bar{\boldsymbol{\sigma}}_t, & \text{if } t \mod 2 = 0 \\ -z^{max}\bar{\boldsymbol{\sigma}}_t, & \text{if } t \mod 2 = 1 \end{array} \right\}. \tag{3.12}$$

We observe that when the sign alternates, ALIE is unable to accumulate error throughout training, which leads to normal convergence. We illustrate the convergence behaviors of ALIE and ALIE with alternating sign of $\mathbf{\Delta}$ in Fig. 3.1 against the TM aggregator, which is an aggregator that is known not to be robust against ALIE [Baruch et al., 2019]. This comparison verifies our argument that the ALIE's strength mainly comes from the

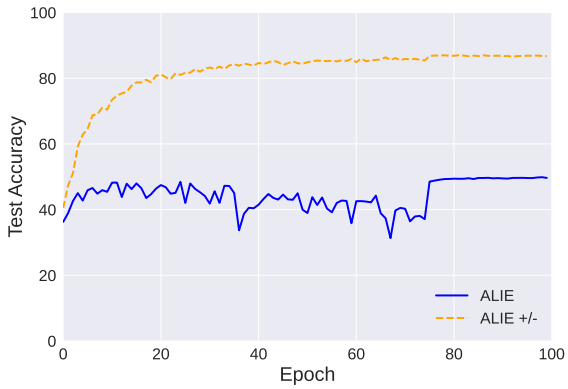


Figure 3.1: ALIE attack on TM aggregation, ALIE (blue) refers to the standard ALIE, while ALIE $+/-$ (orange) is the version of ALIE where the sign of the perturbation alternates at each iteration.

accumulation of attacks over iterations due to the temporal alignment.

3.4 Structuring the attack

Based on the discussions above, for a given radius budget $\|\mathbf{\Delta}_t\| \leq r$, we identify two main design criteria for our attack:

- Keeping attack positively aligned over time; $\max_{\|\mathbf{\Delta}_t\|,\|\mathbf{\Delta}_{t-1}\|\leq r} \cos(\mathbf{\Delta}_t, \mathbf{\Delta}_{t-1})$ to maximize the perturbation accumulated over time.

- Keeping attack orthogonal to the reference update direction, i.e., $cos(\mathbf{\Delta}_t, \tilde{\mathbf{m}}_{t-1}) \approx 0$, which is sufficient to derail the global model, thanks temporal accumulation. On the other hand, unlike IPM, where the poisoned model update is a directly scaled version of the benign update in the opposite direction, i.e., $\alpha\bar{\mathbf{m}}_t$, the attack with orthogonal perturbations is less visible in terms of the angular variation.

3.5 Relocated Orthogonal Perturbation (ROP) Attack

To exploit the aforementioned weaknesses of CC, we introduce a modular and scalable time-coupled model poisoning attack, called *relocated orthogonal perturbation*

Algorithm 3 ROP Attack

Inputs: $\tilde{\mathbf{m}}_{t-1}$, $\bar{\mathbf{m}}_t$, π, λ, ρ

1: $\mathbf{p} \leftarrow \mathbf{1} \in \mathbb{R}^d$

2: **if** $t == 1$ **then**

3: $\quad \tilde{\mathbf{m}}_{t-1} \leftarrow \bar{\mathbf{m}}_t$

4: $\hat{\mathbf{m}}_t \leftarrow \lambda\tilde{\mathbf{m}}_{t-1} + (1-\lambda)\bar{\mathbf{m}}_t$ # target point

5: $\tilde{\mathbf{p}} \leftarrow Proj(\mathbf{p}, \mathbf{m}_t)$

6: $\hat{\mathbf{p}} \leftarrow \mathbf{p} - \tilde{\mathbf{p}}$ # orthogonal perturbation

7: $\mathbf{\Delta}_t \leftarrow \sin(\pi)\frac{\hat{\mathbf{p}}}{||\hat{\mathbf{p}}||} + \cos(\pi)\frac{\hat{\mathbf{m}}_t}{\|\hat{\mathbf{m}}_t\|}$ # direction of the attack

8: $\mathbf{m}_t^{attack} \leftarrow z\mathbf{\Delta}_t + \rho\tilde{\mathbf{m}}_{t-1} + (1-\rho)\bar{\mathbf{m}}_t$ # relocation

(ROP). The proposed attack consists of two main steps; forming an orthogonal perturbation with respect to a target vector and relocating the perturbation, possibly closer to the reference point used by the robust aggregator, in order to avoid norm-based defence mechanisms. First, the attacker picks a point between $\tilde{\mathbf{m}}_{t-1}$ and $\bar{\mathbf{m}}_t$ as the target, denoted by $\hat{\mathbf{m}}_t$ (Algorithm 3 line 4) using the λ hyper-parameter. We emphasize that by using both $\tilde{\mathbf{m}}_{t-1}$ and $\bar{\mathbf{m}}_t$, we induce a certain temporal correlation between $\hat{\mathbf{m}}_t$, thanks to $\tilde{\mathbf{m}}_{t-1}$, but also take into account the current model update.

Once the attack decides on the target $\hat{\mathbf{m}}_t$, an orthogonal vector $\hat{\mathbf{\Delta}}_t$ is formed by using the vector projection and rejection methods (Algorithm 3, lines 5-6). Next, the perturbation is generated as a linear combination of the generated orthogonal vector $\hat{\mathbf{p}}$ and the target $\hat{\mathbf{m}}_t$, so that one can play with the angle between the generated perturbation and the target point using the π hyper-parameter between [0-360] (Algorithm 3, line 7). In the final step, the objective is to relocate the perturbation towards the reference point $\tilde{\mathbf{m}}_{t-1}$ in order to escape norm-based clipping strategies used to sanitize model updates as in CC.

We remark that, though CC takes advantage of $\tilde{\mathbf{m}}_{t-1}$ to sanitize local updates before aggregation, once the described attack, successfully avoids sanitation, it acts as a catalyst for poisoning the CC aggregator, since $\tilde{\mathbf{m}}_t$, used as a center for clipping in the next iteration, also becomes poisoned and unreliable. However, if the attacker knows the aggregator and defences deployed by the PS, such as index-wise aggregator like TM, similar to the ALIE, the attacker can relocate the perturbation back to $\bar{\mathbf{m}}_t$ using the ρ hyper-

parameter (Algorithm 3 line 8). However, accumulating the perturbation to the $\tilde{\mathbf{m}}_{t-1}$ is almost equally effective according to our simulations. In the next section, we will design an even stronger attack that can also employ side information for the network architecture itself.

Chapter 4

SPARSE BYZANTINE ATTACKS

This section highlights the key design aspects of the sparse Byzantine attack method, providing the intuition behind each of its components. Before introducing the details, we revisit the ALIE attack, which will be the base for our design due to certain important features we identify.

4.1 Revisiting ALIE From the Defence Perspective

The key design objective of the ALIE is to find the adversarial update vectors, $\mathbf{m}_{adv}$, such that: 1) they are not statistical outliers, and 2) they are strong enough to shift the aggregate value. To achieve these objectives simultaneously, ALIE utilizes the index-wise statistics, i.e.,

$$\mathbf{m}_{adv} = \bar{\mathbf{m}} - z\bar{\boldsymbol{\sigma}}, \tag{4.1}$$

where $\bar{\boldsymbol{\sigma}}$ is the vector of index-wise standard deviation values and z^{max} is the scaling factor adjusted according to the Byzantine ratio for imperceptibility. From the construction, one can easily observe how ALIE can escape from index-wise statistical examinations and the corresponding defence methods such as TM.

As discussed in [Baruch et al., 2019], ALIE is also effective against defence mechanisms that utilize geometric distance as a metric to identify outliers, such as Krum, where the aggregation rule selects a model update from the available model updates, including both clean and Byzantine updates, which achieves the minimum value of the neighborhood distance measure.[1] We emphasize the following observation: the structure of both

[1]Multi-Krum relies on the same measurement; however, instead of performing an aggregation rule that directly chooses a representative model, Multi-Krum first eliminates the outliers and then performs averaging as an aggregation rule for the remaining ones.

Krum and TM defence mechanisms are based on outlier elimination, and even if they successfully detect the Byzantines, they also falsely classify some benign updates as outliers, which is more frequent when the data is distributed in a non-iid fashion. In [Karimireddy et al., 2022], it has been further shown that these types of defence mechanisms are self-damaging; in the sense that, they deteriorate the performance even when there are no Byzantines, or if the Byzantines just mimic the benign clients as an attack.

In order to avoid falsely discarding benign clients, sanitization-based defences first sanitize all the model updates to minimize the impact of Byzantines and then perform an aggregation over all the sanitized models. Indeed, as we later numerically illustrate through extensive simulations, the CC framework, which sanitizes models through vector-wise clipping, is quite effective against the ALIE attack. However, as mentioned in 3, we can identify certain limitations of CC as well. In particular, CC only scales the deviation from the reference model (aggregate momentum in the previous round); and thus, it is angular invariant to the direction of the perturbation. When a Byzantine client follows ALIE strategy, it sends $\bar{\mathbf{m}} - z\bar{\boldsymbol{\sigma}}$ as a poisoned model update. The CC mechanism first measures the drift with respect to the reference update $\tilde{\mathbf{m}}_t$;

$$\boldsymbol{\Delta}_t^{ref} = \tilde{\mathbf{m}}_t - \bar{\mathbf{m}}_t + z\bar{\sigma}_t, \tag{4.2}$$

then clip the drift $\boldsymbol{\Delta}_t^{ref}$ if its norm is larger than τ, i.e.,

$$\tilde{\boldsymbol{\Delta}}_t^{ref} = \min\left\{1, \frac{\tau}{||\boldsymbol{\Delta}_t^{ref}||}\right\} \boldsymbol{\Delta}_t^{ref}, \tag{4.3}$$

which we refer to as the *effective perturbation*, and as one can observe, the clipping only depends on the norm of $\boldsymbol{\Delta}_t^{ref}$. Hence, when ALIE utilizes a larger scaling parameter z, CC can detect and clip the attack. However, such clipping does not necessarily change the direction of the adversarial perturbation nor prevents their correlation over time, which enhances the time-coupled nature of ALIE, as illustrated in 3, and plays a key role in diverging the model over many iterations.

To clarify the discussion above, we conduct an experiment where we employ ALIE with scaling parameters $z \in \{0.25, 0.5, 1, 1.5\}$ and CC with $\tau = 1$ as a robust aggregator. Then, for each scaling value we measure the average norm of the drift $||\boldsymbol{\Delta}_t^{ref}||$, the expected angle between the effective perturbation $\tilde{\boldsymbol{\Delta}}_t^{ref}$ and the reference update $\tilde{\mathbf{m}}_t$, denoted

Table 4.1: ALIE attack with different scaling values (z) against the CC aggregator on IID CIFAR-10 dataset trained on Resnet-20 architecture with $k = 25$ and $k_m = 5$.

z	**Angle**	**Cos. Sim.**	**Norm**	**Accuracy (%)**
0.25	92	0.82	0.4	82
0.5	88	0.93	0.7	66
1	77	0.97	1.2	50
1.5	76	0.98	1.7	50

by $\angle(\tilde{\mathbf{\Delta}}_t^{ref}, \tilde{\mathbf{m}}_t)$, and finally the temporal correlation of the effective perturbation over iterations, measured by the cosine similarity between the consecutive effective perturbations, $\cos(\tilde{\mathbf{\Delta}}_t^{ref}, \tilde{\mathbf{\Delta}}_{t-1}^{ref})$. The results are illustrated in Table 4.1, which clearly demonstrates that being clipped by the CC does not imply being sanitized properly. As the scaling parameter z increases, CC starts clipping the perturbation $\mathbf{\Delta}_t^{ref}$, as the norm goes beyond $\tau = 1$. But under such clipping, the orthogonality of the perturbation is still preserved, and as highlighted by 3.5, this is quite important for derailing the model, and results in the Table 4.1 show that the accuracy of the model drops sharply as z reaches 1, clearly illustrating the inefficiency of CC in preventing the attack. The key observation here is that, as the scaling parameter z increases, the time-coupled nature of ALIE, measured through cosine similarity, becomes more and more visible. Thus, we conclude that the CC strategy gives a false sense of security against ALIE, and its performance highly depends on the choice of z.

Next, we move our focus to the robust aggregators that utilize index-wise outlier investigation, such as CM and TM, and analyze their performance with respect to the scaling parameter $z \in \{0.25, 0.5, 1, 1.5, 2\}$. For the analysis, we again consider the same simulation setup, but this time we measure the final test accuracy and the escape ratio, which is the average percentage of the indices where the Byzantine model evades the index-wise statistical outlier investigation; in the case of CM, this means the median is equal to the poisoned model. The results are exhibited in Table 4.2. One immediate observation from the results is that the best performance is observed when $z = 0.5$, and increasing the value

of z leads the attack to be more visible and as a result, less effective. Indeed in [Baruch et al., 2019] the authors have stated: "We would like to induce directed small changes to many parameters, instead of large changes to a few", whereas we will show in this paper that both can be performed at the same time.

4.2 Our Key Design Aspects:

Based on the above discussion and observations, we argue that the ultimate objective should not always be avoiding detection, and the attack can seek a better trade-off between being invisible and inducing a stronger perturbation. The more interesting observation, which inspired our proposed Byzantine attack is that, in terms of the final test accuracy, effectively attacking only a certain portion of the indices but with a larger perturbation might be more effective than attacking all the indices with relatively smaller perturbations. This is in contrast to the objective of [Baruch et al., 2019] as highlighted by the quote above. To achieve this goal, we will utilize a hybrid design that combines two different types of attacks, one is more conservative and imperceptible and the other is more aggressive but less imperceptible, which can be obtained simply by choosing two different scaling z values. We will further explain in detail in Section 4.3, how these two sub-attacks can be generated and combined to form our attack.

Until this point, we have shown that when a sufficiently large z value is chosen, both CC and index-wise defence methods, TM and CM, fail against ALIE. We next illustrate that a larger z is not desirable against geometric-distance based defence strategies like Bulyan and Krum. We again consider the same simulation setup, with Bulyan and Krum as the robust aggregators, and the results are illustrated in Table 4.3. We emphasize that, unlike the previous robust aggregators, with Bulyan and Krum, we observe a drastic change in accuracy when the value of z goes beyond a certain threshold; in our simulation, it appears at $z = 1.5$. In contrast to our previous observations, in the case of geometric distance-based defences, when Byzantines are spotted, they immediately become ineffective since, unlike in index-wise investigation, these defences consider all the indices, either Byzantine or benign. This observation further supports our claim regarding the need for a hybrid attack. By attacking aggressively, with a larger z, to only a certain

Table 4.2: Index-wise escape ratios for the ALIE attack with different scales against the CM and TM aggregators. Acc. corresponds to IID CIFAR-10 test accuracy trained on Resnet-20 with $k = 25$ and $k_m = 5$.

z	CM		TM	
	Escape ratio (%)	**Acc. (%)**	**Escape ratio (%)**	**Acc. (%)**
0.25	66	45.43	99.9	72.7
0.5	14	39.4	98	36.35
1	0	48.3	63	47.9
1.5	0	47.5	25	45.1
2	0	49.3	7.5	42.9

Table 4.3: Escape ratios for ALIE attack with different scaling values (z) agaisnt the Multi-Krum and Bulyan aggregators. Acc. corresponds to IID CIFAR-10 test accuracy trained on Resnet-20 with $k = 25$ and $k_m = 5$.

z	**Krum**			**Bulyan**		
	Client%	**Indices %**	**Acc. %**	**Client %**	**Indices %**	**Acc. %**
0.25	100	-	55.75	100	73	47.13
0.5	100	-	45.15	100	60	47.43
1	100	-	45.32	100	29	32.79
1.5	0	-	87.4	0	0	86.54

subset of the indices, the imperceptibility of the attack against geometric-distance based investigation can be preserved. In the next subsection, we formalize the structure of the proposed hybrid attack based on the discussions above.

4.3 Hybrid Sparse Attack (HSA) Design

Now the objective is to introduce a Byzantine attack framework that can escape from both index-wise and Euclidean distance-based sanitization methods. In the previous section, we highlighted that although index-wise statistically imperceptible attacks, e.g., ALIE, are effective against index-wise defence strategies, such as TM, thanks to their time cou-

pling nature, they often become impotent against Euclidean distance-based defence mechanisms, such as CC. Based on these observations, we propose a hybrid Byzantine attack that can be decomposed into two parts; each part targets a different type of defence mechanism, i.e.,

$$\boldsymbol{\Delta}_t = \boldsymbol{\Delta}_{1,t} + \boldsymbol{\Delta}_{2,t}, \tag{4.4}$$

where $\boldsymbol{\Delta}_{1,t}$ is formed by following a similar strategy to ALIE, that is, the poisoned update $\bar{\mathbf{m}}_t + \boldsymbol{\Delta}_{1,t}$ will be statistically imperceptible to index-wise inspection with respect to $\bar{\mathbf{m}}_t$, i.e.,

$$\bar{\mathbf{m}}_t[i] + \boldsymbol{\Delta}_{1,t}[i] \in (\bar{\mathbf{m}}_t[i] - \alpha\sigma_{i,t}, \bar{\mathbf{m}}_t[i] - \alpha\sigma_{i,t}). \tag{4.5}$$

Despite being bounded, adversarial perturbation $\boldsymbol{\Delta}_{t,1}$ accumulates over time to inhibit convergence. On the other hand, for $\boldsymbol{\Delta}_{2,t}$, we focus on an Euclidean distance-based constraint, i.e.,

$$||\boldsymbol{\Delta}_{2,t}||_2 << ||\bar{\mathbf{m}}_t||_2. \tag{4.6}$$

Hence, when $\boldsymbol{\Delta}_{2,t}$ is employed alone, it can easily escape defences that utilize geometric distance as an anomaly metric, and it cannot be detected by angular inspection either.

The key challenge here is how to preserve the aforementioned features of each attack when they are linearly combined to form the final adversarial perturbation $\boldsymbol{\Delta}_t$. To this end, we consider the partiton of te model indices $[d]$ into two complementary sets $\mathcal{M}_1$ and $\mathcal{M}_2$ where $\mathcal{M}_1 \bigcup \mathcal{M}_2 = [d]$ and $\mathcal{M}_1 \cap \mathcal{M}_2 = \emptyset$, and set $\boldsymbol{\Delta}_{t,2}[i] = 0$ if $i \in \mathcal{M}_1$ and $\boldsymbol{\Delta}_{t,1}[i] = 0$ if $i \in \mathcal{M}_2$. Further, by choosing $|\mathcal{M}_1| << d$, we can assure (4.5) holds for most of the indices, hence still achieving index-wise imperceptibility. Accordingly, we generate the Byzantine perturbation $\boldsymbol{\Delta}_t$ in the following way: We first generate a sparse binary mask $\mathbf{c} \in \{0, 1\}^d$ (later we will further explain how we obtain such a mask), and then generate a pair of scaling parameters (z_1, z_2) which are used to form the adversarial perturbation in the following way

$$\boldsymbol{\Delta}_t = (z_1(1 - \mathbf{c}) + z_2\mathbf{c}) \odot \boldsymbol{\sigma}_t, \tag{4.7}$$

and then the poisoned model is simply obtained by adding $\boldsymbol{\Delta}_t$ to the benign update $\bar{\mathbf{m}}_t$. Hence, the key design problem regarding the sparse Byzantine attack is generating the

sparsity mask $\mathbf{c}$ and finding the proper pair of scaling values (z_1, z_2). Assume that $\mathbf{c}$ is known for now. Since from the construction, we expect the part corresponding to $\mathbf{\Delta}_{t,1} = z_1(1-\mathbf{c})$ to be imperceptible against index-wise investigation, we set $z_1 = z_1^{max}$, that is the maximum scaling value obtained with respect to the Byzantine ratio as introduced in ALIE.

We have already highlighted the trade-off between the imperceptibility and the strength of perturbation; thus, choosing an arbitrarily large z_2 value may not be effective. According to Chebyshev's inequality, we have:

$$Pr(|\bar{\mathbf{m}}[i] - \mathbf{m}[i]| > z_2\sigma[i]) \leq \frac{1}{z_2^2}. \tag{4.8}$$

Hence, any choice of $z_2 > \sqrt{2} \approx 1.5$ would make the perturbation highly visible from a statistical perspective; indeed this argument is highly aligned with the initial numerical experiments, in which I investigate the visibility of the perturbation as illustrated in Table 4.2 and Table 4.3 with respect to various values of z, where we observe that scaling beyond $z = 1.5$ makes the attack to be easily spotted by known defence mechanisms.

Lastly, we also note that rather than fixing $z_1 = z_1^{max}$, it is possible to seek an adaptive strategy such that the optimal value of z_1 is reset at each iteration with respect to the benign updates $\mathbf{M}_t \triangleq \{\mathbf{m}_{i,t};\}_{i\in\mathcal{K}_b}$, as previously argued in [Shejwalkar and Houmansadr, 2021]. Once z_2 is fixed, z_1 can be set dynamically at each iteration by solving an optimization problem to avoid geometric-distance-based outlier detection, similar to the optimization-based attack design in [Shejwalkar and Houmansadr, 2021]. To be more precise, one can obtain $z_1^\star$ by maximizing scaling factor z_1, under the average benign neighbourhood distance constraint

$$z_1^\star = argmax_z \underbrace{\frac{1}{k_b}\sum_{i\in\mathcal{K}_b} ||\bar{\mathbf{m}}_t - z(1-\mathbf{c})\odot\boldsymbol{\sigma}_t - z_2\mathbf{c}\odot\boldsymbol{\sigma}_t - \mathbf{m}_{i,t}||}_{D_{avg}(\mathbf{m}_t,\mathbf{c},\boldsymbol{\sigma}_t,z,z_2)} \leq \Delta_{th}, \tag{4.9}$$

where,

$$\Delta_{th} = \frac{1}{k_b - 1}\left(\max_{i\in\mathcal{K}_b}\sum_{j\in\mathcal{K}_b} ||\mathbf{m}_{i,t} - \mathbf{m}_{j,t}||\right). \tag{4.10}$$

The approximate solution to the aforementioned optimization problems can be obtained through an iterative search algorithm by changing step-size [Shejwalkar and Houmansadr,

Algorithm 4 Hybrid Sparse Attack

for $t = 1, \ldots, T$ **do**
  if t == 1 **then**
    Initialization Inputs: Byzantine datasets $\mathcal{D}_i \in \mathcal{K}_m$, target sparsity δ, pruning policy Π, initial model weights $\boldsymbol{\theta}_0 \in \mathbb{R}^d$
    Initialization Phase:
    Generate mask $\mathbf{c}$ for given $\mathcal{D}, \Pi, \boldsymbol{\theta}_0, \delta$
  Inputs: Benign updates $\mathbf{m}_t$, benign statistics $(\boldsymbol{\sigma}_t, \bar{\boldsymbol{m}}_t)$, z_1^{max}, z_2^{max}
  Generate pair of scaling parameters $(z_1^{(t)}, z_2^{(t)})$
  Option 1: Fixed policy
  Use fixed scaling $z_1^{(t)} = z_1^{max}$ and $z_2^{(t)} = z_2^{max}$
  Option 2: Semi-adaptive policy
  Use fixed $z_2^{(t)} = z_2^{max}$ and optimize $z_1^{(t)}$ acc. Min-Sum Policy [Shejwalkar and Houmansadr, 2021]
  Generate perturbation: $\boldsymbol{\Delta}_t = (z_1^{(t)}(1 - \mathbf{c}) + z_2^{(t)}\mathbf{c}) \odot \boldsymbol{\sigma}_t$
  Poisoned model: $\mathbf{m}_{adv} = \bar{\mathbf{m}}_t - \boldsymbol{\Delta}_t$

2021]. We note that when $\mathbf{c} = \mathbf{1}_d$, the proposed attack reduces to the one introduced in [Shejwalkar and Houmansadr, 2021], hence the design in [Shejwalkar and Houmansadr, 2021] is a special case of proposed Byzantine framework where we search for optimal z_1, z_2, and $\mathbf{c}$ instead of a single z parameter. At this point, we also note that the design in [Shejwalkar and Houmansadr, 2021] is oblivious to the NN architecture, whereas, as we further discuss later, the proposed attack utilizes the NN architecture being attacked, which will be reflected in $\mathbf{c}$. Finally, as we illustrate later, the proposed solution works well even if both scaling parameters z_1 and z_2 are fixed, which may be more practical from an implementation perspective. The overall hybrid sparse Byzantine attack mechanism is presented in Algorithm 4.

4.4 Generating Sparse Mask

In the previous part, we explained how one can generate a Byzantine attack for a given sparsity mask $\mathbf{c}$. In this section, investigate how to generate a sparsity mask $\mathbf{c}$. The vanilla

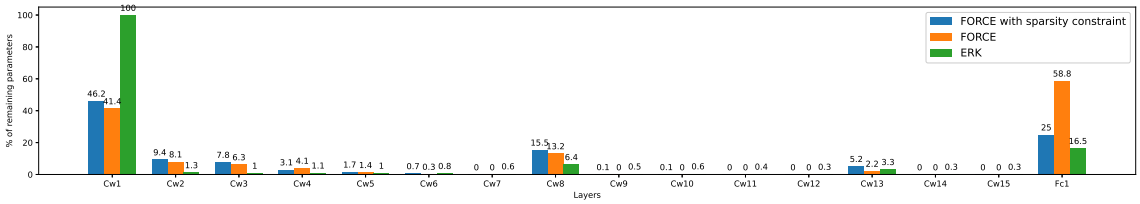


Figure 4.1: Distribution of the non-sparse locations (remaining weights) in ResNet-20 architecture after pruning with ERK, Vanilla FORCE, and FORCE with a sparsity constraint on the fully-connected penultimate layer, $\delta_{FC}^{max} = 0.25$, for $\delta = 0.005$. The x-axis denotes the NN layers, where Cwi denotes the weights of the ith convolutional layer, and Fc1 denotes the weights of the FC layer at the end.

approach to generate a sparsity mask is random sparsity; that is, for a given sparsity ratio δ, we can randomly generate a binary value for each index i.e., for each $i \in [d]$,

$$\mathbf{c}[i] = \begin{cases} 0, & \text{if } w.p.\ \delta \\ 1, & \text{if } w.p.\ 1-\delta \end{cases} \tag{4.11}$$

The previous studies on neural network pruning have shown that, for a moderate sparsity level δ, random sub-networks often achieve the performance of dense networks [Frankle and Carbin, 2019]. Thus, by establishing an analogy between a sparse attack and a sparse network, we argue that a random sparsity pattern, that is, setting the number of NN weights being attacked at each layer as proportional to the dimension of that layer, would be a good candidate for a sparse Byzantine attack. Indeed, as we later demonstrate, a random sparsity mask works well against various defence mechanisms.

However, regarding the robustness of the trained model, the impact of each layer may not be identical, and from the network pruning perspective, one may need to assign a different sparsity ratio to each layer. To clarify, for a set of L layers, one may consider a different sparsity ratio δ_l that satisfies the overall sparsity constraint $\sum_{l\in\mathcal{L}} \delta_l d_l = \delta d$, where d_l is the number of parameters in layer l, i.e., $\sum_{l\in\mathcal{L}} d_l = d$ In the literature, there are different approaches to distribute the sparsity among layers; for instance, in [Evci et al., 2020], the authors utilize *Erdos-Renyi-Kernel (ERK)* formulation.

Critical Layers: Another important aspect often emphasized in the literature on sparse networks is the existence of sensitive layers, which are excluded from the network pruning

since these parts of the NN model are highly sensitive [de Jorge et al., 2021, Tanaka et al., 2020, Vogels et al., 2019]. These critical layers are often batchnorm layers and bias parameters and we use $\mathcal{L}_{critical}$ to denote the set of critical layers. From the perspective of Byzantines, similar to the idea of keeping these layers dense in network pruning, we argue that the Byzantine attack should particularly target these sensitive layers. Accordingly, we let all the critical layers $l \in \mathcal{L}_{critical}$ kept dense, that is all the corresponding values in $\mathbf{c}$ are set to one.

We remark that the approaches that have been highlighted above *are omniscient to the target dataset* and do not induce a high *computational complexity*. On the other hand, it has been known that, in high-sparsity regimes, randomly generated sub-networks do not perform well. Equivalently, if we want to design an attack with a high sparsity pattern, we can further utilize a small portion of the training data, available to the colluding Byzantines, and perform network pruning to identify those important weights to be attacked. Further discussions on the aforementioned approaches, as well as different sparsity ratios, will be presented in the next section. In Section 4.6, we will revisit some of the network pruning strategies that work well with limited or no data, to form the sparse attack. Overall, results show that the use of a network pruning strategy has a clear advantage compared to a randomly generated sparsity.

4.5 Network Prunning

In this section, we further provide the details regarding how network pruning works and the implementation of the FORCE network pruning framework.

4.5.1 Sparsity Scheduler

Similar to [de Jorge et al., 2021], we consider exponential decay sparsity scheduling that is over T prunning steps the number of non-sparse positions decreased from $\kappa_0 = d$ to $\kappa_T = \kappa$ gradually in the following manner:

$$\kappa_t = \lfloor e^{(\frac{t}{T}\log\kappa + (1-\frac{t}{T})\log d)} \rfloor. \tag{4.12}$$

Algorithm 5 Iterative Network Prunning

Inputs: Target sparsity κ, Initialized weights $\boldsymbol{\theta} \in \mathbb{R}^d$, Time steps T, Local dataset $\{D_i\}_{i\in\mathcal{K}_m}$
Outputs: $\mathbf{c}$
Initialize $\mathbf{c}_0 \leftarrow \mathbf{1}_d$
Scheduled Sparsity $\{\kappa_1, \ldots, \kappa_T\}$ acc. eqn. (4.12)
for $t = 1, \ldots, T$ **do**,
    for $i \in \mathcal{K}_m$ **do**
        Sample mini-batch: $\zeta \in D_i$
        Compute Saliency: $\mathbf{s}_i(\boldsymbol{\theta}, \mathbf{c}_{t-1}) = |\boldsymbol{\theta} \odot \nabla F(\mathbf{c}_{t-1} \odot \boldsymbol{\theta}; \zeta)|$
    Consensus Saliency: $\mathbf{s}(\boldsymbol{\theta}, \mathbf{c}_{t-1}) \leftarrow \frac{1}{k_m} \sum_{i\in\mathcal{K}_m} \mathbf{s}_i(\boldsymbol{\theta}, \mathbf{c}_{t-1})$
    Indices with maximum saliency measure $\mathcal{I} = Top_{\kappa_t}(\mathbf{s})$
    Update binary mask: $\mathbf{c}_t \leftarrow \mathbf{c} : \mathbf{c}[i] = 1, \text{ if } i \in \mathcal{I}$

4.5.2 Prunning Algorithm

The overall iterative pruning algorithm, executed over T steps with respect to the sparsity scheduling in 4.12, is illustrated in Algorithm 5, where at each iteration t, each byzantine node first computes the saliency $\mathbf{s}_i(\boldsymbol{\theta}, \mathbf{c}_{t-1})$, with respect to initial network weights $\boldsymbol{\theta}$ and the precious binary sparsity mask $\mathbf{c}_{t-1}$ according to (4.12), then seek a consensus on the saliency measure. Then, based on the saliency score, the non-sparse locations for the given scheduled sparsity constraint κ_t, simply through a Top_k operation.

4.6 Sparsity with Network Pruning

The overall objective of NN pruning is to find a small subset of NN parameters $\mathcal{M}$ with size $||\mathcal{M}|| = \kappa << d$ such that the NN model with only the subset of the parameters $\boldsymbol{\theta}_{\mathcal{M}}$ is sufficient to achieve a certain loss over the given dataset $\mathcal{D}$. Formally speaking, the objective can be formulated as

$$\begin{aligned} \min_{\mathbf{c},\boldsymbol{\theta}} F(\mathbf{c} \odot \boldsymbol{\theta}; \mathcal{D}) &= \tfrac{1}{|\mathcal{D}|} \textstyle\sum_{\zeta\in\mathcal{D}} \quad f(\mathbf{c} \odot \boldsymbol{\theta}; \zeta) \\ &\text{s.t. : } ||\mathbf{c}||_0 \leq \kappa \text{ and } \boldsymbol{\theta} \in \mathbb{R}^d. \end{aligned} \tag{4.13}$$

In our work, we mainly focus on NN pruning strategies that are performed with a limited dataset since the Byzantines may have access to a limited number of data samples to form

a sparsity pattern. In SNIP [Lee et al., 2019], the authors highlight that, for a given $\boldsymbol{\theta}$, the additional loss incurred due to the pruning of the i^{th} weight is given by,

$$\Delta F_i = F(\mathbf{1} \odot \boldsymbol{\theta}; \mathcal{D}) - F((\mathbf{1} - \mathbf{e}_i) \odot \boldsymbol{\theta}; \mathcal{D}) \tag{4.14}$$

where $\mathbf{e}_i$ is a one-hot vector where the i^{th} value is one and zeros everywhere except the i^{th} index. It can be approximated by gradient with respect to the $\mathbf{c}$:

$$\Delta F_i \approx \frac{\partial F(\mathbf{c} \odot \boldsymbol{\theta}; \mathcal{D})}{\partial c_i}\bigg|_{\mathbf{c}=\mathbf{1}} = \tag{4.15}$$
$$lim_{\delta \longrightarrow 0} \frac{F(\mathbf{c} \odot \boldsymbol{\theta}; \mathcal{D}) - F((\mathbf{c} - \delta \mathbf{e}_i) \odot \boldsymbol{\theta}; \mathcal{D})}{\delta}\bigg|_{\mathbf{c}=\mathbf{1}}.$$

The term above implies a directional derivative, a sparse direction; and thus, can be written in the following form:

$$\boldsymbol{\theta}_i \frac{\partial F(\boldsymbol{\theta}; \mathcal{D})}{\delta \boldsymbol{\theta}_i}. \tag{4.16}$$

We can conclude that $s_i \triangleq |\boldsymbol{\theta}_i \frac{\partial F(\boldsymbol{\theta}; \mathcal{D})}{\boldsymbol{\theta}_i}|$ can be used as a saliency measure for the connection sensitivity [Lee et al., 2019]. Though building on the same idea, FORCE [de Jorge et al., 2021] argues for measuring the connection sensitivity *after* pruning rather than before. Hence, FORCE evaluates the gradient for the foresight sparse model:

$$\frac{\partial F(\mathbf{c} \odot \boldsymbol{\theta}; \mathcal{D})}{\partial \mathbf{c}}\bigg|_{\mathbf{c}=\hat{\mathbf{c}}}, \ ||\hat{\mathbf{c}}||_0 = \kappa, \hat{\mathbf{c}} \in \{0, 1\}^d . \tag{4.17}$$

Accordingly, FORCE searches for the binary sparse mask $\mathbf{c}$ that maximizes the following saliency measure:

$$S(\boldsymbol{\theta}, \mathbf{c}) = \sum_{i \in [d]: \mathbf{c}[i]=1} \underbrace{|\boldsymbol{\theta} \odot \nabla F(\mathbf{c} \odot \boldsymbol{\theta}; \mathcal{D})|}_{\triangleq \mathbf{s}(\boldsymbol{\theta}, \mathbf{c})}[i] , \tag{4.18}$$

where $\nabla F(\mathbf{c} \odot \boldsymbol{\theta}; \mathcal{D}) = \left(\frac{\partial F(\bar{\boldsymbol{\theta}}; \mathcal{D})}{\partial \bar{\boldsymbol{\theta}}}\big|_{\mathbf{c}}\right)$, $\bar{\boldsymbol{\theta}} = \mathbf{c} \odot \boldsymbol{\theta}$, from equation (5) in [de Jorge et al., 2021]. However, a strategy that measures connection sensitivity after pruning is practically not feasible due to its combinatorial complexity; that is, ideally, we need to compute the saliency of all possible $\binom{d}{\kappa}$ sparsity patterns. To mitigate the combinatorial complexity, FORCE proposes to perform pruning in an iterative manner, under the assumption that

the change of the gradient can be neglected, rather than checking each possible sparsity pattern individually. Please for refer to [de Jorge et al., 2021] the details.

In our design, we utilize the FORCE framework since it does not require the whole dataset, which is aligned with our problem setup, where we assume Byzantines can access only a small portion of the dataset. Although there is a strong analogy between NN pruning and sparse Byzantine attacks, when we implement the FORCE framework for high sparsity regime, such as $\delta \leq 1\%$, we observe that non-sparse indices are accumulated in certain layers of the network, e.g., early convolutional layers and penultimate fully-connected (FC) layer for the convolutional neural networks (CNN) architectures. In Fig 4.1, we plot (orange) the percentage of the remaining (non-pruned) weights for each layer of the ResNet-20 architecture after pruning with FORCE. From an attack perspective, targeting only certain layers may not be effective, especially if the employed robust aggregator performs a layer-wise investigation [Varma et al., 2021, Liu et al., 2023]. Therefore, we modify the FORCE framework by imposing certain layer-wise sparsity constraints so that the corresponding attack is more homogeneously distributed among the layers; in other words, we aim to prevent certain layers from being overemphasized. Accordingly, we introduce the term δ_{max} to impose that the sparsity ratio of a layer cannot be larger than δ_{max}. Our experiments show that pruning methods often overemphasize certain layers, particularly the penultimate FC layer, as illustrated in Fig. 4.1. Hence, we argue that imposing constraints for specific layers is often sufficient for our purpose. The sparsity pattern obtained by setting $\delta_{FC}^{max} = 0.25$ is illustrated in Fig. 4.1 (in color blue), where we observe that the penultimate FC layer becomes more sparse the number of non-sparse positions in early convolutional layers increases.

Chapter 5

SEQUENTIAL CENTERED CLIPPING (S-CC) DEFENCE

In Section 3.5, we have shown that the predictability of the reference point can be exploited by the Byzantines to reconfigure their attack strategy based on the knowledge of the reference point. In this section, we argue that it is possible to enhance the robustness of the aggregators, particularly CC, by hiding the reference point from the Byzantines. Accordingly, we introduce the idea of inducing randomness in the selection of the reference point, in contrast to the static one used in the original CC framework.

To overcome the aforementioned vulnerabilities of the CC and to defend against the proposed model poisoning attack such as ROP, we propose an enhanced version of CC, named sequential CC (S-CC), given in Algorithm 6. The main idea of S-CC is to divide the clients into disjoint groups and then perform CC and aggregation sequentially over each group so that the reference point utilized for each group is different and depends on the previous groups, which makes it hard for Byzantines to collude and predict the reference point. Furthermore, it reduces the variance among the clients by taking the average of the groups [Karimireddy et al., 2022]. The key difference from our proposed bucketing approach is that we employ cosine similarity to sort and cluster the clients based on their similarity to the reference point before applying it to bucketing. Whereas [Karimireddy et al., 2022] just randomly distributes clients to buckets and [Allouah et al., 2023] employs Nearest Neighbor Mixing, which essentially employs Multi-Krum [Blanchard et al., 2017] to form the buckets based on their geometric similarities. However, due to the nature of the Krum, it is a considerably slower bucketing approach than our proposed approach and random bucketing [Karimireddy et al., 2022].

Algorithm 6 Sequential centered clipping (S-CC)

Inputs: $\tilde{m}_{t-1}$, $\{\mathbf{m}_{i,t}\}_{i\in\mathcal{K}}$, $f_{CC}(\cdot)$, τ

1: Determine number of buckets $R \leftarrow \lceil k/n \rceil$

2: Form buckets of size n, $C_1, \ldots, C_R$ by selecting one client from each cluster w.o. repetition.

3: Initialize auxiliary reference momentum: $\hat{\mathbf{m}}_t = \tilde{\mathbf{m}}_{t-1}$

4: **for** $n = 1, \ldots, R$ **do**

5: $\quad \bar{\mathbf{m}}_t = \frac{1}{\|C_n\|} \sum_{i\in C_n} \tilde{\mathbf{m}}_{i,t}$

6: $\quad \bar{\mathbf{m}}_t = f_{CC}(\bar{\mathbf{m}}_{i,t} | \hat{\mathbf{m}}_t, \tau)$

7: $\quad \hat{\mathbf{m}}_t = \bar{\mathbf{m}}_t$

8: $\tilde{\mathbf{m}}_t = \hat{\mathbf{m}}_t$

5.1 Design

Key motivation behind S-CC is to introduce randomness into the CC framework. This is achieved by grouping the clients into buckets of size n, while performing CC in an iterative manner, instead of a single aggregation with a fixed reference point. S-CC performs the aggregation in $\lceil k/n \rceil$ consecutive phases while dynamically updating the reference point at the end of each phase to induce certain randomness to prevent the collusion of the Byzantine clients. Here, n is a static hyper-parameter chosen by the PS before the training starts (Alg. 6, line 1). At the beginning of each aggregation step, the clients are sorted by the PS based on the cosine similarity between their momentums and the reference point and grouped into n clusters (line 3). PS randomly selects one client from each cluster without repetition to form a bucket and performs CC to update the momentum using the average momentum of the bucket (Alg. 6, lines 4-6). After clipping the average value of each bucket, the reference point of S-CC is updated (Alg. 6, line 7). Therefore, unlike in standard CC, momentum is updated $\lceil k/n \rceil$ times sequentially while also reducing the number of clipping operations. One weakness of the S-CC aggregator is the decrease of robustness when there is a Byzantine in multiple clusters, which is more apparent in attacks like ALIE, where Byzantines target the $\bar{\mathbf{m}}_t$. In such attacks, cosine similarity between the reference of the S-CC and the momentum of the Byzantine can vary depending on the variance among benign clients, which can result in multiple clus-

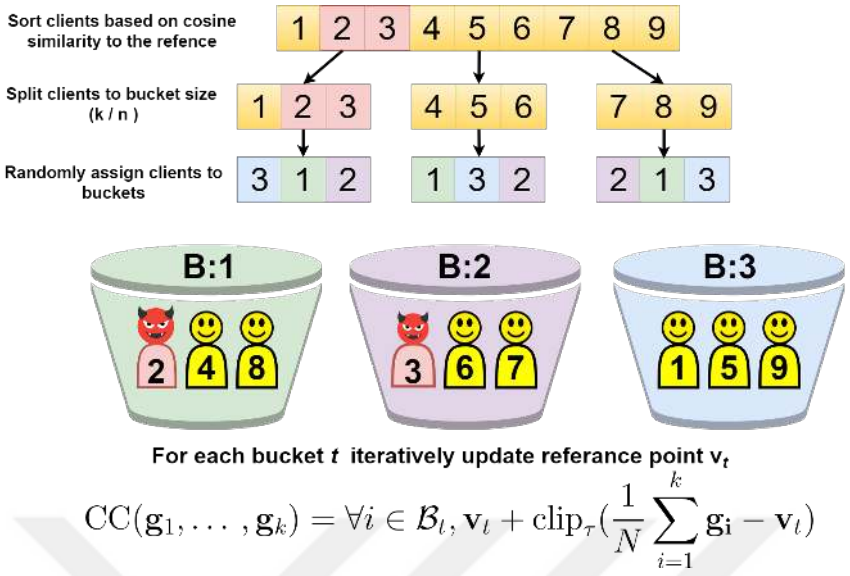


Figure 5.1: Representation of S-CC with $k = 9$ and $k_m = 2$ using bucket size of $n = 3$.

ters with Byzantines. Consequently, some buckets may contain more than one Byzantine, resulting in partial collusion. To prevent this, we recommend using S-CC with local momentum employed, more specifically $\beta = 0.9$, to ensure lower variance among the clients, which results in higher cosine-similarity between $\bar{\mathbf{m}}_t$ and $\tilde{\mathbf{m}}_{t-1}$. Overall S-CC framework is illustrated in 5.1

Finally, although the proposed S-CC strategy has certain similarities with the iterative CC strategy introduced in [Karimireddy et al., 2021] and the bucketing strategy introduced in [Karimireddy et al., 2022], the way we utilize clustering and multi-phase aggregation methods significantly differs from those and aims to address a particular vulnerability of the CC mechanism. Iterative CC is introduced in [Karimireddy et al., 2021] as a refined version of CC, where clipping is performed in a successive manner to eventually converge to a true update over a certain number of iterations in order to achieve a better sensitivity for the clipping by updating reference point in each iteration and thus momentum's of the clients are clipped multiple iterations in a single aggregation step. The proposed S-CC strategy differs from iterative CC in two main design aspects: First, unlike the iterative CC, not all the clients are present at each iteration of S-CC. Performing CC iteratively over groups induces randomness in the reference points. This aims to prevent Byzantines from

accurately predicting reference points and avoid clipping by relocating the perturbation accordingly. Second, by utilizing a systematic grouping strategy using cosine similarity based clustering and then bucketing, which is not considered in iterative CC, S-CC aims to minimize the potential collusion among the Byzantines, since a different reference point is employed for each group.

Chapter 6

EXPERIMENTAL RESULTS

6.1 Analysis on ROP and local momentum

6.1.1 Simulation Setup

We consider synchronous FL with a total of k=25 clients. We assume 20% of the clients, i.e., k_m=5 are Byzantine, which is in line with [Karimireddy et al., 2021]. For training, we follow a similar setup as in [Karimireddy et al., 2021], where we train our neural networks for 100 epochs with a local batch size of 32 and an initial learning rate of $\eta = 0.1$, which is reduced at epoch 75 by a factor of 0.1. For all the simulations, each local client considers the cross entropy loss to compute gradients.

6.1.2 Datasets and Networks

For the grayscale image classification task, FMNIST [Xiao et al., 2017] dataset and train a convolutional neural network (CNN) which consists of 2 convolutional layers with 20 and 50 filters, respectively, and 1 fully connected layer. For the RGB image classification task, we consider CIFAR-10 and CIFAR-100 datasets [Krizhevsky et al.,], and train ResNet-20 and ResNet-9 architectures, respectively. Since CIFAR-100 is a relatively challenging dataset for classification, we only consider the IID scenario for this task.

6.1.3 Adversary and FL model

We consider synchronous FL with a total of k=25 clients. We assume 20% of the clients, i.e., k_m=5 of are Byzantine, which is in line with [Karimireddy et al., 2021]. For training, we follow a similar setup as in [Karimireddy et al., 2021], where we train our neural networks for 100 epochs with a local batch size of 32 and an initial learning rate of $\eta = 0.1$, which is reduced at epoch 75 by a factor of 0.1. For all the simulations, each local client

considers the cross entropy loss to compute gradients.

For simulations with CC, we set its radius to $\tau = \{0.1, 1\}$ and the number of clipping iterations to $l = 1$. We observe that CC is more prone to divergence when $\tau \geq 10$, and since the authors in [Karimireddy et al., 2021] argue that every τ is equally robust, we only consider these two τ values. We consider a bucket size of $n = 3$ for the proposed S-CC aggregator.

For omniscient model poisoning attacks, we consider ALIE, IPM, and ROP. In ROP, we experimentally set $z = 1$, $\lambda = 0.9$, $\pi = 90$ and $\rho = 1$. The impact of z, λ, π, and ρ on the convergence are further discussed in the supplementary materials. For IPM, we use $\epsilon = 0.2$. For non-omniscient attacks, we consider the bit-flip and label-flip attacks [Biggio et al., 2012]. In the bit-flip attack, Byzantine clients flip the signs of their own gradient values, whereas, in the label-flip attack, Byzantine clients flip the label of a sample by subtracting it from the total number of image classes in the dataset (All datasets have even number of classes).

6.1.4 Numerical Resuts

In this section, we empirically demonstrate the effectiveness of our proposed ROP attack against robust aggregators, particularly CC with local momentum. For this, we utilize local momentums of $\beta = [0, 0.9, .99]$. For simulations, we compare ROP with omniscient ALIE [Baruch et al., 2019], IPM [Xie et al., 2020b], and non-omniscient bit-flip and label-flip attacks. In our results, we also report the baseline accuracy of the aggregators, where all the participating clients are benign, i.e., $k_m = 0$. All numerical results are averaged over 5 runs with different seeds.

In Fig. 6.1, we present the effect of ROP and other attacks on the 2-layer CNN architecture trained with. Due to the IID FMNIST dataset's relative simplicity of the task and the robust CNN architecture, most of the aggregators are capable of fending off the Byzantines in this scenario. Only with $\beta = 0$, ALIE can utilize increased variance among the clients, resulting in the divergence of the RFA and TM aggregators. CC is more robust compared to other aggregators; however, ROP can still hinder its convergence, especially when local momentum is employed with $\beta = 0.99$, reducing the test accuracy by 5% and

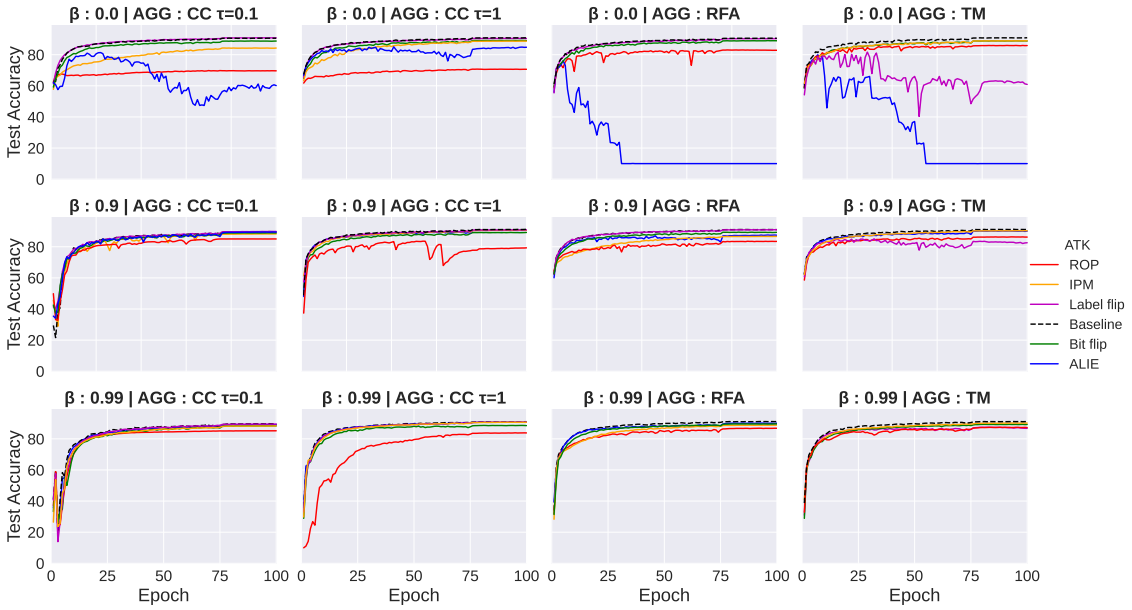


Figure 6.1: 2-layer CNN trained on the fmnist dataset distributed in an IID fashion across the clients. Test accuracy of various aggregation methods are presented in the presence of Byzantines (solid lines).

7% for CC with $\tau = 0.1$ and $\tau = 1$, respectively, while also significantly reducing the convergence speed.

In Fig 6.2, we show the convergence behavior of the ResNet-20 architecture trained on the IID CIFAR-10 dataset. We can observe the effect of time-coupled omniscient attacks like ALIE and ROP. For $\beta = 0$, ALIE benefits from the increased variance, while ROP still surpasses all the attacks against the CC aggregator. Similar to the results reported in [Karimireddy et al., 2021], high local momentum benefits all the aggregators; however, ROP reduces the test accuracy by nearly reducing it by 35% in the case of CC with $\beta = 0.99$, which is the best aggregator as advised by the authors of CC [Karimireddy et al., 2021].

For a more challenging setup of aggregators with IID data distribution, we consider the CIFAR-100 image classification task and train larger ResNet-9 architecture. The results are given in Fig. 6.3. For CIFAR-100, the RFA aggregator struggles to defend against ROP, ALIE, and IPM on every β parameter, while TM can only converge for $\beta = 0.99$, except in the case of an ROP attack. Against the CC aggregator, without employing a local

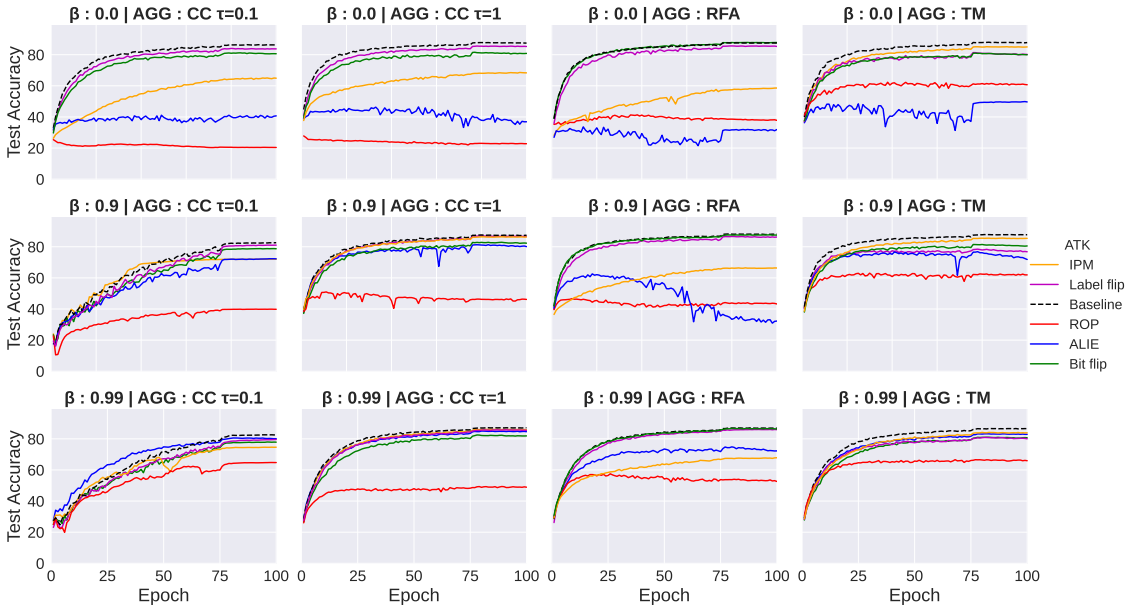


Figure 6.2: ResNet-20 trained on the CIFAR-10 dataset distributed in an IID fashion across the clients. Test accuracy of various aggregation methods are presented in the presence of Byzantines (solid lines).

momentum, time-coupled attacks are capable of derailing the convergence at a certain point, while ROP is capable of hindering the learning process from the beginning of the training. Similar to the CIFAR-10 results, only ROP can prevent convergence consistently when local momentum is employed; it can reduce the test accuracy by 40% for $\beta = 0.99$.

In Fig. 6.4, we experimented with non-IID distribution on the FMNIST dataset. According to this simulation, ALIE is able to diverge the RFA aggregator, while ROP and IPM are able to yield sub-optimal accuracy. Interestingly, against the TM aggregator, the non-omniscient label-flip attack is the most successful when local momentum is employed. Overall, CC with $\tau = 1$ and local momentum is the most successful aggregator; however, ROP is still able to slow down its convergence and lower the baseline accuracy by almost 20%.

In Fig. 6.5, we challenge the robust aggregators on the ResNet-20 architecture trained on the CIFAR-10 image classification task with non-IID data distribution. In terms of baseline accuracy, i.e., without any attack, CC with $\tau = 0.1$ with local momentum fails to converge, while its IID counterpart and other aggregators can provide normal convergence

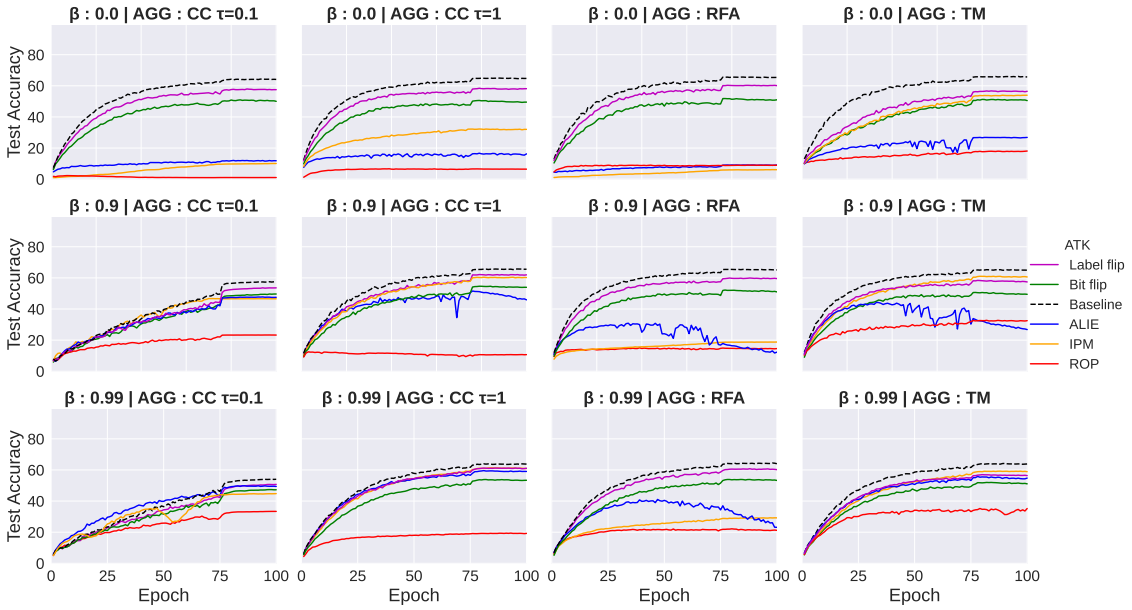


Figure 6.3: ResNet-9 trained on the CIFAR-100 dataset distributed in an IID fashion across the clients. Test accuracy of various aggregation methods are presented in the presence of Byzantines (solid lines).

when there is no Byzantine client. Against all the aggregators and β values, ROP is capable of preventing the convergence from the start of training; meanwhile, with the benefit of the increased variance due to non-IID data distribution, ALIE is also a strong competitor to ROP, especially when local momentum is not employed. However, ALIE assumes to know the variance; thus, increased variance greatly helps, meanwhile, ROP does not assume such knowledge yet still surpasses the ALIE. In this scenario, TM with $\beta = 0.99$ surpasses the CC aggregator in terms of robustness, although ROP can still reduce its test accuracy by 35%.

Overall, we show that ROP overcomes the robustness of CC regardless of the τ and β values, and it is the most successful attack in the IID data scenarios, especially when local momentum is employed. Other attacks fail to prevent the convergence of the CC aggregator with $\tau = 1$. In the Non-IID scenario, ALIE is also a successful attack as ROP. This is mainly due to the increased variance among clients, which provides ALIE more room to scale up its perturbation, while ROP does not assume to know the variance, and uses the same amount of perturbation regardless of the variance among benign clients.

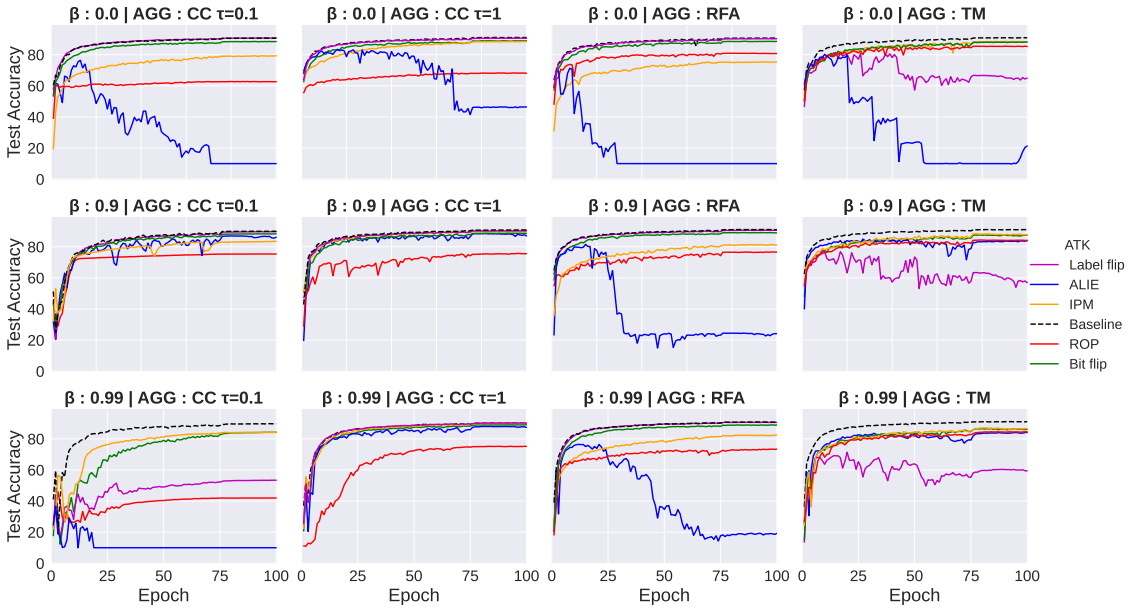


Figure 6.4: 2-layer CNN trained on the fmnist dataset distributed in a non-IID fashion across the clients. Test accuracy of various aggregation methods are presented in the presence of Byzantines (solid lines).

Although in its default configuration, ROP targets the CC, we use the same configuration on a median-based defence, TM, and a norm-based defence, RFA, both of which consider statistical calculations using only $\mathbf{m}_{i,t}$ from the clients. We observe that ROP can still compete with and even surpass ALIE against these defences. Further analysis of robust aggregators shows that CC with radius $\tau = 0.1$ is not robust when the data is non-IID, and local momentum is employed, failing to converge even when there are no Byzantines. This can result from the combination of low gradient scaling (1-β) and very small CC radius τ, which leads to the PS converging very slowly, or not learning properly from the clients.

6.2 Ablation study on ROP attack

In this section, we further illustrate the effects of the hyper-parameters of ROP, namely, ρ, Π, λ, and z parameters in ROP attack.

For the λ hyper-parameter, we grid search the optimal value between [0, 0.5, 0.9, 1]

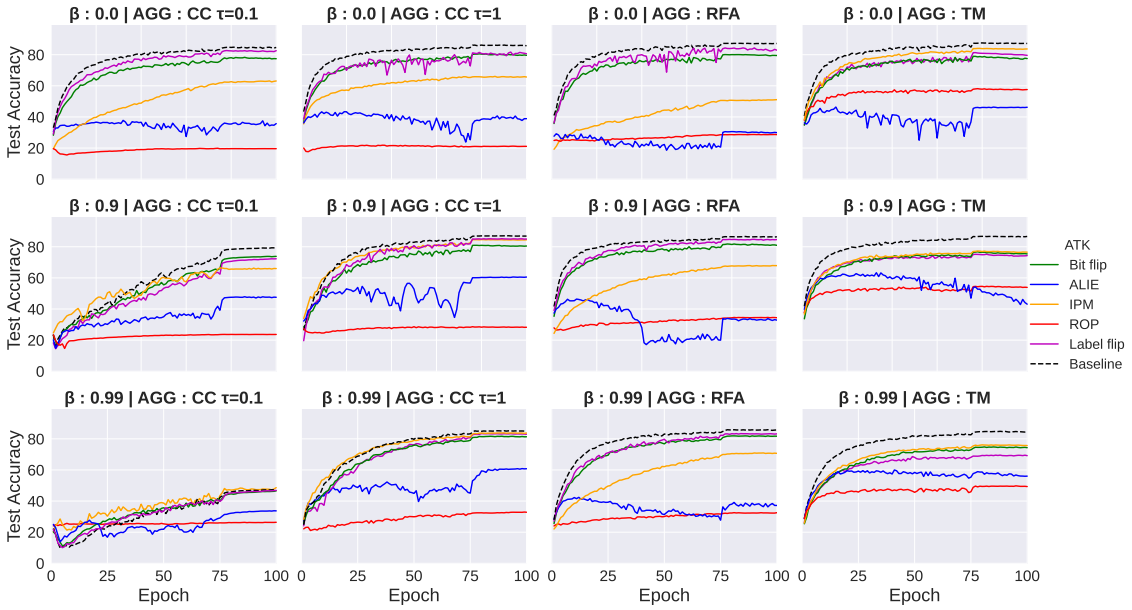


Figure 6.5: ResNet-20 trained on the CIFAR-10 dataset distributed in a non-IID fashion across the clients. Test accuracy of various aggregation methods are presented in the presence of Byzantines (solid lines).

on the IID and non-IID CIFAR10 image classification problem. In Table 6.1, we show that overall $\lambda = 0.9$ results in the strongest attack for multiple aggregators and β values. We find that $\lambda = 1$ is also quite effective to all aggregator types, meaning that Byzantine clients can generate strong attacks without being omniscient by only employing the broadcasted $\tilde{\mathbf{m}}_{t-1}$ from the PS. Furthermore, we analyze the location of the attack and the angle of the attack with ρ and π hyper-parameters, respectively. In our extensive simulation results on Table 6.1, we show that relocating the attack to $\tilde{\mathbf{m}}_{t-1}$ has a greater effect on CC while targeting the $\bar{\mathbf{m}}_t$ can significantly reduce the performance of the RFA. In terms of the angle of the perturbation, $\pi = 90, 120, 135$ are equally capable of diminishing the test accuracy results. By default, ROP employs $\rho = 1$, $\lambda = 0.9$ and $\pi = 90$. For the z hyper-parameter, we grid search the values [1, 10, 100] and find out that all z values are equally robust to the CC aggregator due to the relocation of the attack and angular invariance of the CC. On the TM aggregator, we find that an increased z value increases the robustness of the attack considerably compared to the other aggregators on both IID and non-IID data distributions, which we illustrated in Fig. 6.6 and Fig. 6.7 respectively.

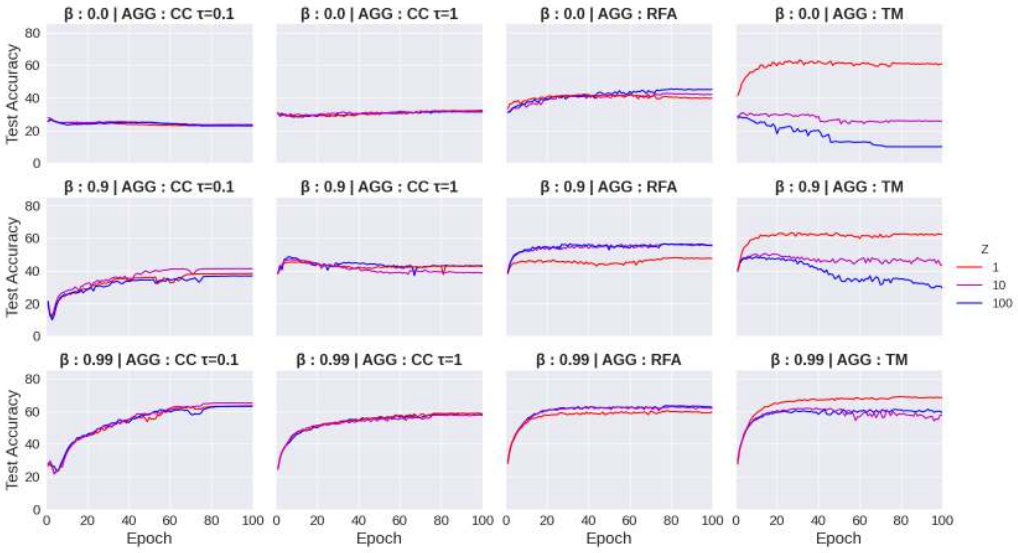


Figure 6.6: Cifar10 IID test accuracy on various aggregation methods against ROP attack with different z values.

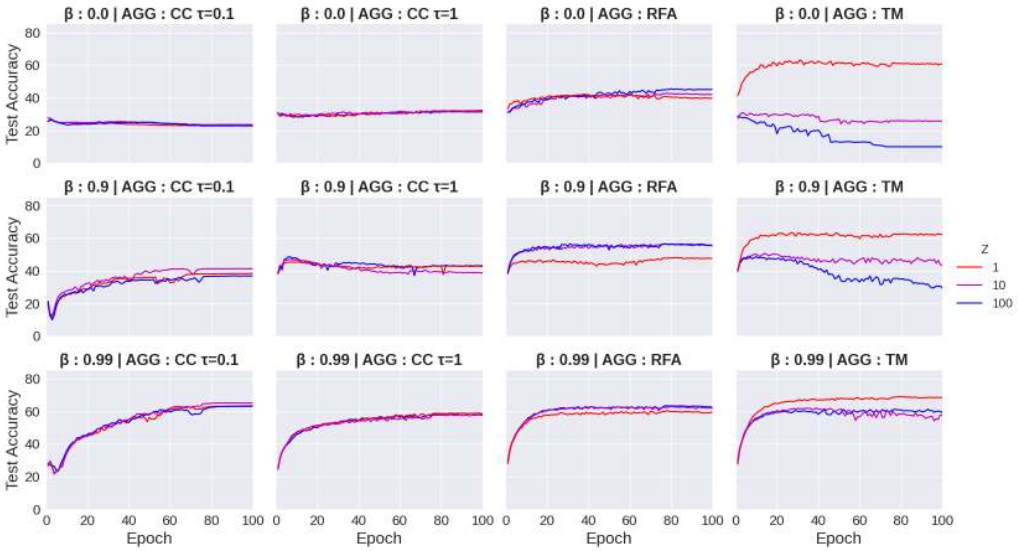


Figure 6.7: Cifar10 non-IID test accuracy on various aggregation methods against ROP attack with different z values.

6.2.1 Training loss analysis

In this section, we show that the proposed ROP attack prevents local clients from reaching local minima by instead converging to saddle points. In Fig 6.8, we show that on IID

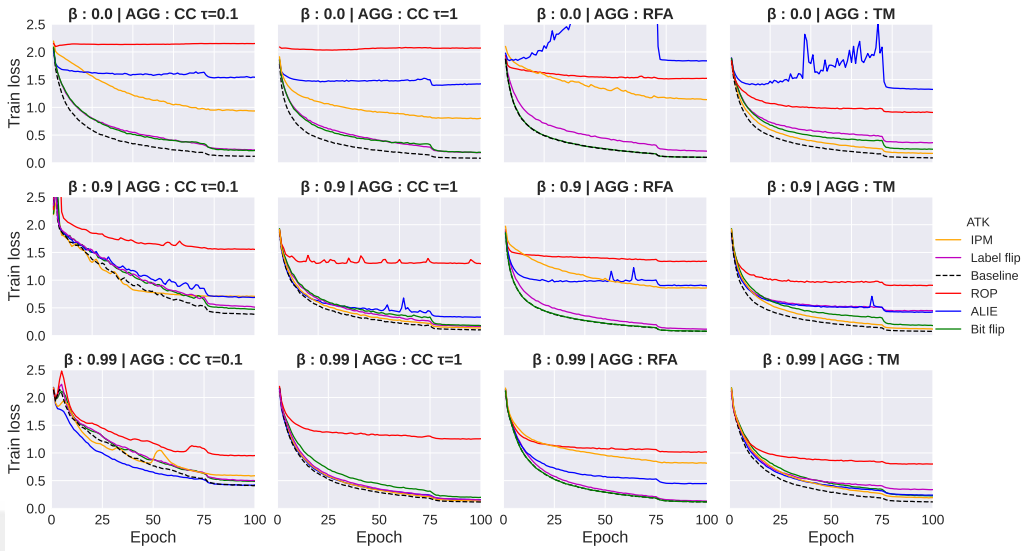


Figure 6.8: Cifar10 IID train loss on various aggregation methods.

distributed CIFAR-10, our ROP attack always converges to a higher training accuracy for all aggregators and β values. Unlike ROP, ALIE converges to saddle points at $\beta = 0$ only, which can be explained by increased variance among the clients when worker momentum is not employed.

6.2.2 Study on total Byzantine and client numbers

Different Byzantine ratios

For this study, we set the total number of clients k=25 with different numbers of Byzantines, specifically k_m=[1, 2, 3, 7, 8, 10, 12] where 12 is the upper bound for the maximum number of Byzantine clients that many aggregators can provide normal convergence. We illustrated our results for IID data distribution in Table 6.2 and non-IID distribution in Table 6.3. We show that the proposed ROP attack can greatly reduce the test accuracy with only k_m=1 and k_m=2, while other attacks struggle to reduce the test accuracy. In Table 6.2, with $\beta = 0$, ROP is capable of reducing the test accuracy between 26-30 %, while on Table 6.3 at $\beta = 0.9$, ROP can reduce the test accuracy between 5-36%, clearly illustrating its effectiveness compared to other attacks. Furthermore, for k_m=[7, 8], ROP increased its effectiveness compared to other attacks. For k_m=[10, 12], we can finally see

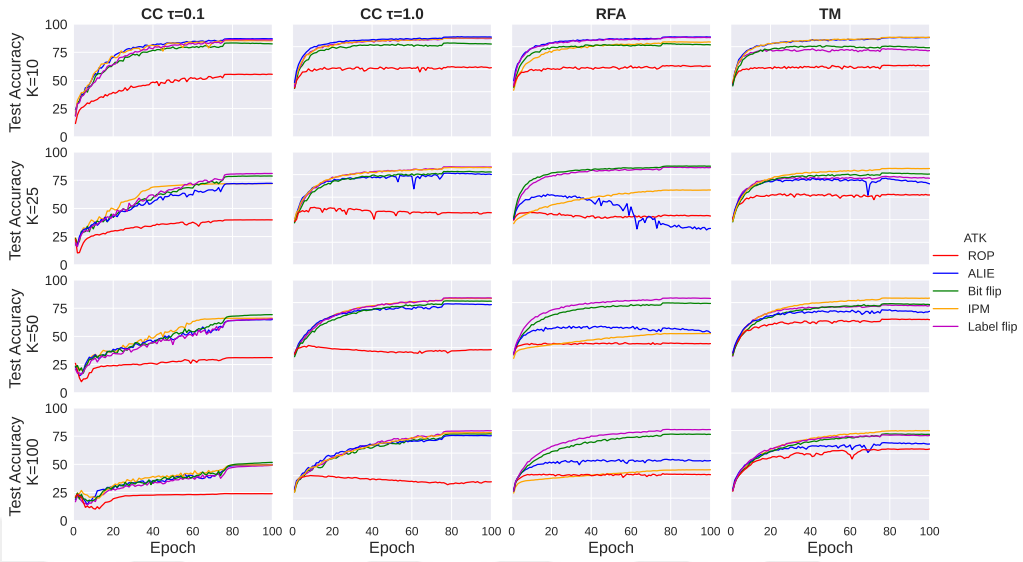


Figure 6.9: CIFAR-10 test accuracy results on IID data at β=0.9. Each row represents the total number of clients k with %20 Byzantines.

the effect of the IPM attack since it requires almost half of the clients as Byzantine to prevent convergence. However, IPM is still not as effective as ROP if local clients employ momentum, as seen in Table 6.2 with β=[0.9, 0.99].

Effect of the total number of clients.

We experimented with k =[10, 25, 50, 100]. For k=10, only the proposed ROP attack can greatly reduce the test accuracy for both IID and non-IID datasets with up to 60% against the CC aggregator, which can be seen in Fig. 6.9 and Fig. 6.10, respectively. For k=[50,100] and non-IID data distributions, ALIE can utilize the higher variance due to the larger number of clients and slightly surpass ROP only against the TM aggregator as seen in Fig 6.10. Furthermore, IPM also benefits greatly from the increased number of clients and is capable of generating attacks that are almost equally effective as ROP and ALIE against the RFA aggregator. Ultimately, for IID-distributed data, the proposed ROP is the most effective attack. In the case of non-IID data, ROP can still overwhelmingly reduce the effectiveness of CC and is almost as equally effective as ALIE, if not surpass it.

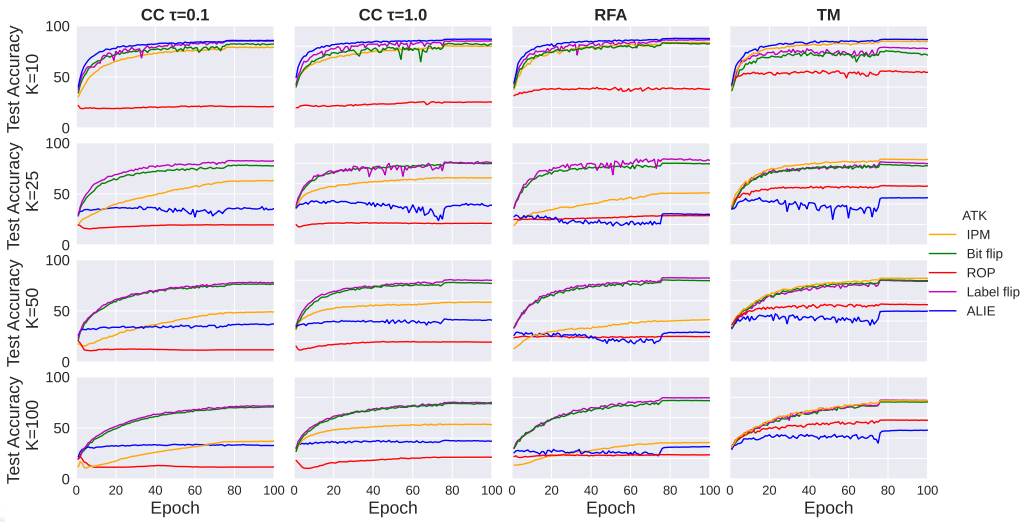


Figure 6.10: CIFAR-10 test accuracy results on non-IID data at β=0.9. Each row represents the total number of clients k with %20 Byzantines.

6.3 Expended experiments with HSA

6.3.1 Simulation Setup

For the simulation setup, we consider a synchronous centralized FL framework with k=25 participating clients and set the Byzantine ratio $\gamma = 0.2$, i.e., k_m=5 clients are malicious, following the same scenario in Section 6.1. In the previous section, we thoroughly go over the effect of the local momentum (β) and why it is critical to Byzantine robustness due to the reduced variance. Ultimately we find that $\beta = 0.9$ is the overall most optimal value regardless of the data distribution and aggregator. As such for this setup we will only consider $\beta = 0.9$.

Datasets and Networks

For the image classification task, we consider MNIST [LeCun and Cortes, 2010], FMNIST [Xiao et al., 2017] and CIFAR-10 [Krizhevsky et al.,] datasets and train them with 2-layer multi-layer perceptron (MLP), 2-layer CNN, and ResNet-20 [He et al., 2016] architectures, respectively. For the data distribution, we consider two scenarios, where the training dataset is distributed among the clients in IID and non-IID manners, respec-

tively. In the IID scenario, we distribute the training data homogeneously among the clients so that each client has an equal number of data for each label. In the non-IID scenario, similarly to existing works [Shejwalkar et al., 2022, Wan and Chen, 2021, Gupta et al., 2022, Yin and Zeng, 2023], the dataset is divided among the clients according to the Dirichlet distribution [Minka, 2000] with parameter α, denoted by $\mathrm{Dir}(\alpha)$, where α parameter controls the skewness of the probability distribution with respect to the class labels. In our experiments, we set $\alpha = 1$, which is argued to model practical scenarios [Shejwalkar et al., 2022]. Aligned with the previous works [Karimireddy et al., 2021], we train the NN architectures for 100 epochs with a local batch size of 32 and an initial learning rate of $\eta = 0.1$, which is reduced at epoch 75 by a factor of 0.1. In all the simulations, we utilize local momentum with $\beta = 0.9$ since the use of local momentum reduces the variance and enhances the robustness against Byzantines.

Aggregators

From the defence perspective, we consider 8 different fundamental robust aggregators namely: Bulyan [El Mhamdi et al., 2018], CC [Karimireddy et al., 2021], CM [Yin et al., 2018], MultiKrum [Blanchard et al., 2017], RFA [Pillutla et al., 2022], SignSGD [Bernstein et al., 2019], and TM [Yin et al., 2018], FL-trust [Cao et al., 2021b]. Further, we also employ a more sophisticated robust aggregation strategy called GAS with two variations, built on Bulyan and MultiKrum respectively [Liu et al., 2023]. Next, we briefly summarize the parameter setup of the robust aggregators. For CC, we employ a fixed ball radius of $\tau = 1$ since a higher radius is more susceptible to Byzantine attacks as we illustrate in 6.1.4 and also according to [Raynal et al., 2023], and lower τ performs suboptimally in non-IID settings and set the number of clipping iterations to 1. For number of sub-vectors in GAS (p), we set $p = 1000$ on ResNet-20 and 2-Layer CNN, while on 2-layer MLP, we set $p = 100$ due to the smaller network size. Our p values are proportional to the best p parameter that authors employed in their works for their NN size [Liu et al., 2023]. For SignSGD, we employ an initial learning rate of 0.01, which makes learning more stable. For FL-Trust, we partition 50 training samples from the original dataset as a root dataset.

Attacks

For non-omniscient attacks, we consider the bit-flip [Li et al., 2019],label-flip [Biggio et al., 2012, Fung et al., 2020, Fang et al., 2020] and customized C&W [Carlini and Wagner, 2017] backdoor attack. In the C&W attack, the attacker flips the label to the most probable non-true label, i.e., the second-best label attack. More formally, $\hat{y} = \text{argmax}(\boldsymbol{\theta}(\mathbf{x}))$, $y \neq \hat{y}$ where $\mathbf{x}$ is input sample, y corresponding label and $\hat{y}$ is the attack label. For the omniscient model poisoning attacks, we consider 3 different attacks: ALIE [Baruch et al., 2019], IPM [Xie et al., 2020b], and ROP. In ROP, we employ the default parameter values. For IPM, previous works have considered different z values, i.e., in [Karimireddy et al., 2021] $z = 0.1$ and in the previous section 6.1.4 $z = 0.2$. In preliminary experiments, we have considered these values, but observed that IPM performs best with $z = 0.4$; hence, in expended results, we report the performance with $z = 0.4$. Apart from the aforementioned non-dynamic attacks, we have also revisited a dynamic attack framework, where the parameters are optimized adaptively over the iterations. In particular, we have implemented Min-sum and Min-max attacks [Shejwalkar and Houmansadr, 2021], utilizing *Inverse standard deviation* as the perturbation vector.

Hybrid Sparse Attacks

In our experiments, we consider four different variations of our hybrid sparse Byzantine attack framework. First, we consider a base strategy $\Pi_r(\delta)$ that generates a random sparse mask with a sparsity ratio of δ, without any constraint; for the second strategy, we impose the constraint that the sparsity ratio of δ is achieved at each layer, which is denoted by $\Pi_{lr}(\delta)$. Finally, for the last two strategies, we modify the first two strategies by imposing the constraint that the critical layers should be non-sparse, i.e., $\delta_l = 1$, for $l \in \mathcal{L}_{critical}$, which are denoted with $\Pi_r^+(\delta)$ and $\Pi_{lr}^+(\delta)$, respectively. In addition to random sparsity, we also consider ERK [Evci et al., 2020] to obtain sparsity patterns for each individual layer, denoted by $\Pi_{ERK}^+(\delta)$, by allocating the sparsity budget among the layers according to the number of incoming and outgoing connections between the hidden layers. Finally, utilize the network pruning-based attack, using the FORCE algorithm, which is denoted by $\Pi_{FC}^+(\delta)$.

Adaptive Hybrid Sparse Attack: Until this point we refer to the hybrid sparse attacks with fixed policy that is the scaling parameters are fixed throughout the training. However, inspired by [Shejwalkar and Houmansadr, 2021], we also consider an adaptive version of our hybrid sparse attack, which is highlighted in Algorithm 3 **option 2**, and referred to as *Hybrid Min-Sum (HMS)*.

6.3.2 Experimental Results

For a fair performance comparison, we consider the existing attacks under two categories based on whether they are dynamically optimized at each round or utilize a fixed policy. For the first category of attacks with fixed policy, we consider bit-flip, label-flip, ALIE [Baruch et al., 2019], IPM [Xie et al., 2020b], and ROP. In the second category, we consider attacks that are optimized at each round such as Min-Sum [Shejwalkar and Houmansadr, 2021] and Min-Max [Shejwalkar and Houmansadr, 2021].

In Table 6.4, we present the final test accuracy results on the CIFAR-10 dataset with IID-distribution and ResNet-20 architecture. We highlight the best attack performance among the fixed, dynamic, and proposed hybrid policies separately, using colors blue, green, and red, respectively. For our proposed Hybrid sparse attack we consider all four variations of the random sparsity masks under four different sparsity regimes, i.e., $\delta = \{5 \times 10^{-3}, 5 \times 10^{-2}, 2 \times 10^{-1}, 5 \times 10^{-1}\}$. In addition to randomly generated sparsity masks, for high sparsity regimes $\delta = \{5 \times 10^{-3}, 2 \times 10^{-1}, \}$, we utilize the network pruning method FORCE to generate sparsity masks with a constraint on the FC layer $\delta_{FC}^{max} = 0.25$ based on our initial discussion that non-sparse values may accumulate at certain layers. We want to remark that, in the scope of this work our main objective is to show the vulnerabilities of the existing defence mechanisms against proposed hybrid sparse attacks and highlight the key design aspects, however, we believe the formation of the sparsity masks can be further investigated to improve the performance of the hybrid sparse attacks by exploring different pruning methods layer-wise sparsity constraints, nonetheless in the scope of this work we have limited our focus. In addition aforementioned hybrid sparse attack variations, we also implement the Hybrid Min-Sum framework which adaptively selects the first scaling parameter while using a fixed one for the sec-

ond.

The corresponding test accuracy results are illustrated in Table 6.4. Before further analysis of the numerical results, we want to remark that both variations of the GAS defence mechanism are indeed highly effective against fixed policy attacks, which highlights the importance of layer-wise investigation of Byzantines. We also observe that, in the case of randomly generated sparsity masks, the best performance is obtained with $\Pi^{+}_{lr}(\delta)$, which induces a layer-wise sparsity constraint and identifies the critical layers prior to generating sparsity mask. The numerical results also indicate that the hybrid sparse attacks with random masks perform the best with a moderate sparsity of $\delta = 2 \times 10^{-1}$, and their strength weakens as we reduce the sparsity ratio. The results also verify the aforementioned design aspects in Section 4, to be more precise as we expected, a hybrid sparse attack demonstrates the most significant improvement, compared to other fixed policy attacks, against Multi-krum, which aim to identify benign updates by performing clustering based on geometric distance, thus more vulnerable to sparse perturbations. On the other hand, as we highlighted in Section 4, those stronger perturbations, compared to ALIE, in sparse locations, are relatively more visible to index-wise examination but also have a stronger impact, which makes the hybrid sparse attack more effective against TM compared to ALIE. As a direct consequence, the hybrid sparse attack is also effective against Bulyan, which is the sequential implementation of TM and M-Krum. On the other hand, we have observed that against RFA, ALIE slightly outperforms hybrid sparse attack[1], nevertheless, we can conclude that, unlike other known attacks, the hybrid sparse attack is effective against all the defence strategies.

To test the advantage of the side information regarding the architecture, and synaptic connectivity, on the performance of the attack, we compare the performance of the attack generated according to $\Pi^{+}_{lr}(\delta)$ and the one generated according to the pruning strategy $\Pi^{+}_{F}(\delta)$, under the sparsity constraint of $\delta = 5 \times 10^{-3}$. Contrary to our initial argument that attacking specific locations, a sub-network obtained through pruning, should enhance the strength of the attack, on average, we do not observe a consistent visible improvement,

[1]Here, we want to emphasize that although ALIE achieves the best results against RFA, it exhibits a high standard deviation.

although in some cases pruning based sparsity mask shows some improvement compared to random masks. However, the interesting observation is that, in the case of CC, which often rescales the adversarial perturbations without changing their directions, the results in high sparsity regime $\delta = 5 \times 10^{-3}$, highlights that $\Pi_F^+(\delta)$, also $\Pi_{ERK}^+(\delta)$, performs significantly better compared attacks with random masks. From this observation and the fact that both ERK and FORCE emphasize certain layers, such as the FC layer and early convolutional layers, we argue that the network architecture can be utilized to identify the critical layers to increase the impact of the attack. However, since the impact of neural pruning-based masks are limited to CC defence, we argue that such strategies may overemphasize certain layers, and the advantage becomes the Achilles' heel since the accumulation of attack at certain layers may make an attack to relinquish its imperceptibility against other defences.

To verify the argument above, we follow a strategy where we again utilize the FORCE algorithm to perform network pruning, but induce an additional constraint on the FC layer, i.e., $\delta_{FC}^{max} = 2.5 \times 10^{-1}$, in order to achieve a sparsity mask, in which non-sparse positions are distributed more uniformly compared to the previous case. Under this new sparsity constraint, for sparsity ratios $\delta = 5 \times 10^{-3}$ and $\delta = 1 \times 10^{-2}$, we observe visible improvements in the performance, especially against the GAS defence mechanism, which divides the flattened NN weights into smaller sub-vectors and performs aggregation for each subvector separately. We also remark that, FORCE with δ_{FC}^{max} outperforms ERK-based sparsity masks since, similar to the FORCE, which overemphasizes the initial convolutional layer and particularly the penultimate FC layer, the ERK method overemphasizes the initial convolutional layer, where all the weights are preserved as illustrated in Fig. 4.1 with the green color. Hence, the initial convolutional layer being overemphasized makes the ERK method less effective against certain defence mechanisms, such as GAS.

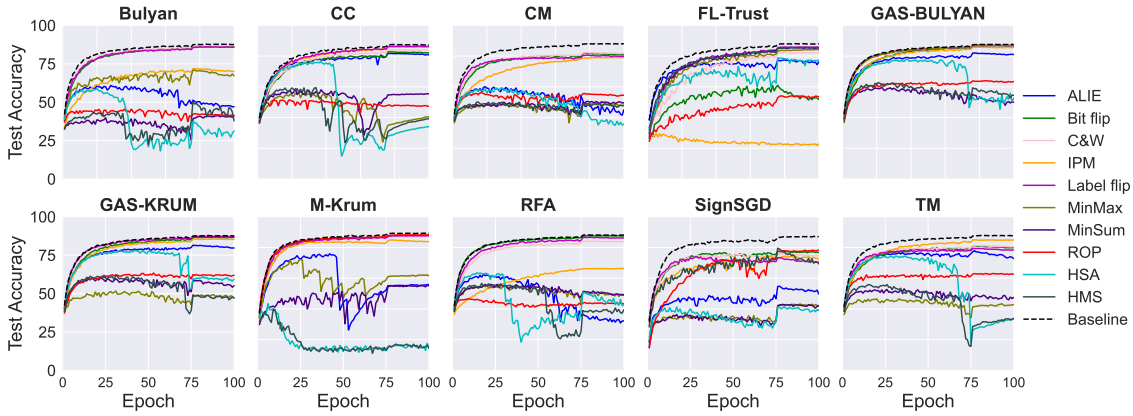
Overall, from the discussion above, we conclude that there is a certain trade-off regarding the formation of the attack; the underlying sparsity mask should be designed in parallel to the topology-based connectivity and sensitivity patterns identified through network pruning, but also the position of the non-sparse weights, where the attack targets more aggressively, should be not be concentrated at particular layers to avoid being perceptible. As we already discussed above, such unintended concentration can be mitigated

by enforcing a layer-wise sparsity constraint. Through extensive simulations, we have analyzed various alternatives for regularizing the layer-wise sparsity, and the best yet simple solution is to have a constraint for the FC layer with δ_{FC}^{max} when the FORCE method is used to generate a sparsity mask.

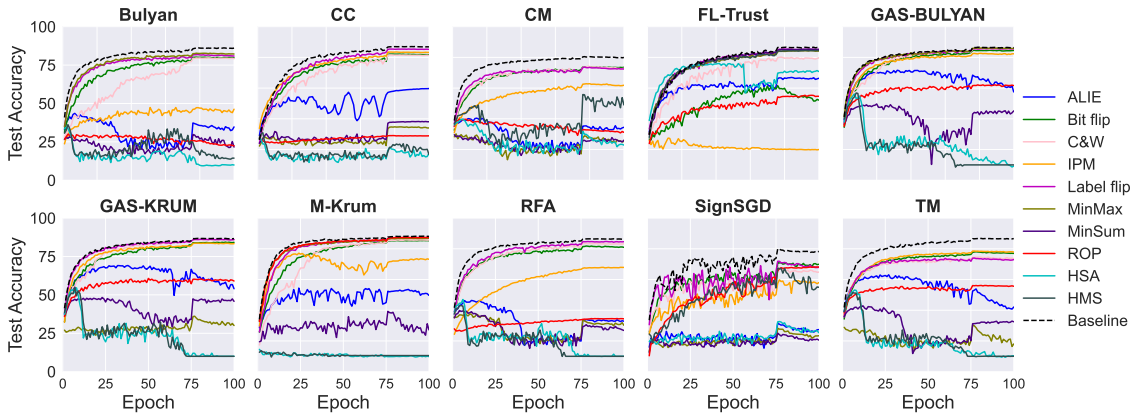
Finally, we test the adaptive hybrid sparse attack, Hybrid Min-Sum, which utilizes the sparsity mask generated by the FORCE method for sparsity ratio $\delta = 5 \times 10^{-3}$ and $\delta_{FC}^{max} = 2.5 \times 10^{-1}$. Our experimental results show that Hybrid Min-Sum either outperforms Min-Sum and Min-Max or exhibits comparable performance. Especially, against particular defence mechanisms such as M-KRUM, CC, RFA, and TM, utilization of the architectural knowledge significantly improves the performance of the Min-Sum method.

We repeat the same simulations for the non-IID data distribution scenario, and the results are presented in Table 6.5. In this more practical scenario, we have shown that the proposed hybrid sparse attack overcomes all the known defence mechanisms, including GAS, which is highly effective against all fixed policy Byzantine attacks. In this scenario, GAS achieves a minimum test accuracy of %60 against fixed policy Byzantines, and even achieves attack-free accuracy for Bit Flip, IPM, and Label Flip. However, we have shown that our proposed attack $\Pi_F^+(\delta)$ with $\delta_{FC}^{max} = 2.5 \times 10^{-1}$ significantly reduces the test accuracy, to be more precise, against all defence mechanism our proposed attack able to keep test accuracy below %25 and make the model to converge in most of the cases. One of the key aspects here is that the performance of $\Pi_{FC}^+(\delta)$, compared to its baseline form ALIE, demonstrates how attacking only a very limited number of carefully selected weights enhances the strength of the attack significantly. Consequently, the simulation results obtained for ALIE, Min-Sum, and hybrid sparse attack, where all three strategies utilize the same base attack formation, demonstrate that the level of aggressiveness and the carefully chosen target locations significantly increase the strength of the attack, which particularly visible in the case of GAS defence.

Additionally in Fig. 6.11, we illustrate the convergence of the all considered attack and robust aggregators, where 6.11a and 6.11b correspond to IID and non-IID simulations respectively. In Fig 6.11, we consider the $\Pi_F^+(\delta_1, \delta_{FC}^{max})$ as HSA. Both HSA and HMS are effective for all aggregators and data distributions except the FL-Trust where ROP, IPM, and Bit-Flip can surpass the HSA and HMS. However, similar to the ROP, if the attacker



(a)



(b)

Figure 6.11: Test accuracy results of training ResNet-20 architecture with CIFAR-10 dataset distributed IID (6.11a) and non-IID(6.11b) over $k = 25$ clients $k_m = 5$ of them being malicious. Training and testing are performed over 100 epochs and repeated for 10 different aggregation mechanisms under 10 Byzantine attack strategies. The reported results are obtained by averaging 3 independent trials.

changes to attack location to the reference of the aggregator $\hat{\mathbf{m}}_t$, both HSA and HMS can reduce the test accuracy up to % 50.

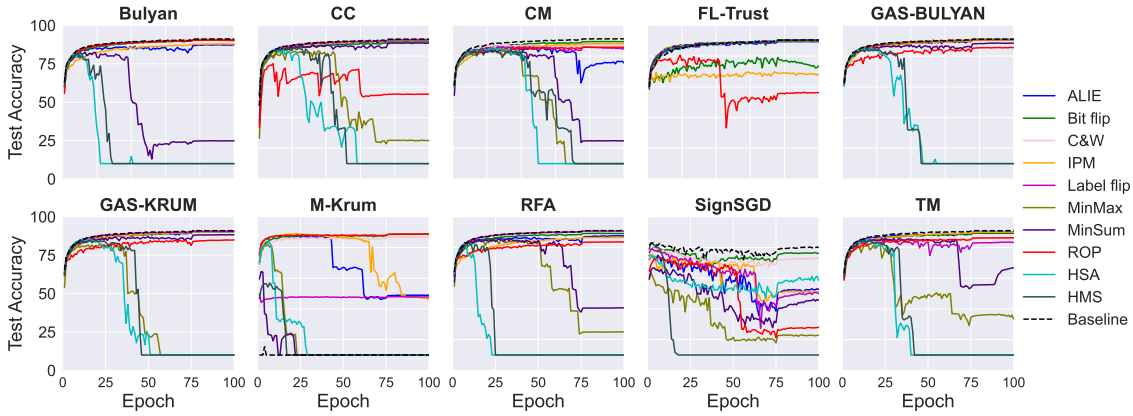
Since smaller NN architectures are more vulnerable to Byzantine attacks, we mainly focus on moderately large NN architectures, i.e., ResNet-20, and the classification task on CIFAR-10. For the sake of completeness, in this subsection, we further conduct the same experiments with different NN networks and datasets, to be more precise, we have analyzed the byzantine performance and compare the existing attack and defence frameworks for the classification task on the FMNIST dataset with a 2-layer CNN architecture and classification task on MNIST dataset with 2-layer MLP architecture. For our hybrid sparse attack (HSA), we consider the configuration of $\Pi_F^+(\delta = 5 \times 10^{-3}, \delta_{FC}^{max} = 0.25)$ which often provides the best results in the case of CIFAR-10 dataset except the MNIST where we do not induce any limitation since MLP network only consist of fully connected layers as such we utilize $\Pi_F^+(\delta = 1 \times 10^{-2})$ for MNIST classification task. The experimental results and the convergence behavior under different robust aggregators and Byzantine attacks are illustrated in Fig. 6.12 and Fig. 6.13, respectively.

On the F-MNIST classification objective, HSA is capable of diverging the global model on both IID and non-IID simulations except on the SignSGD aggregator, where it reduces test accuracy by 35% on IID and 55% on non-IID simulations as illustrated in Fig 6.12b.

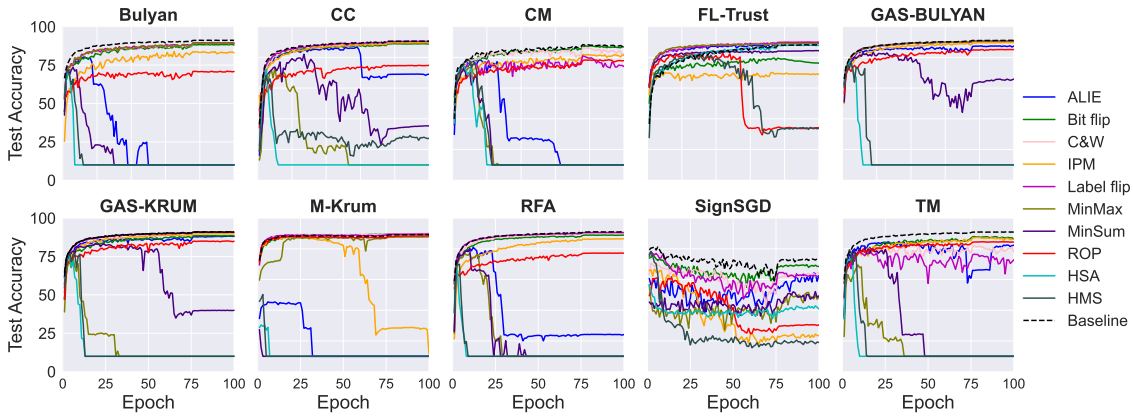
For the MNIST classification task, HSA only failed at the CC aggregator with IID simulations and failed to achieve the best attack performance in the CM aggregator on non-IID simulations while diverging the PS model every other simulation as illustrated in the Fig. 6.13.

6.4 S-CC Defence Results

In Fig. 6.14, we show that the proposed S-CC aggregator is capable of increasing the test accuracy against all attack types regardless of the data distribution, while employing no local momentum, i.e., $\beta = 0$. The S-CC aggregator is equally robust for both IID and non-IID data distributions, which is not the case for other aggregators we have considered in our simulations. Therefore, the S-CC aggregator is robust to all model poisoning attacks

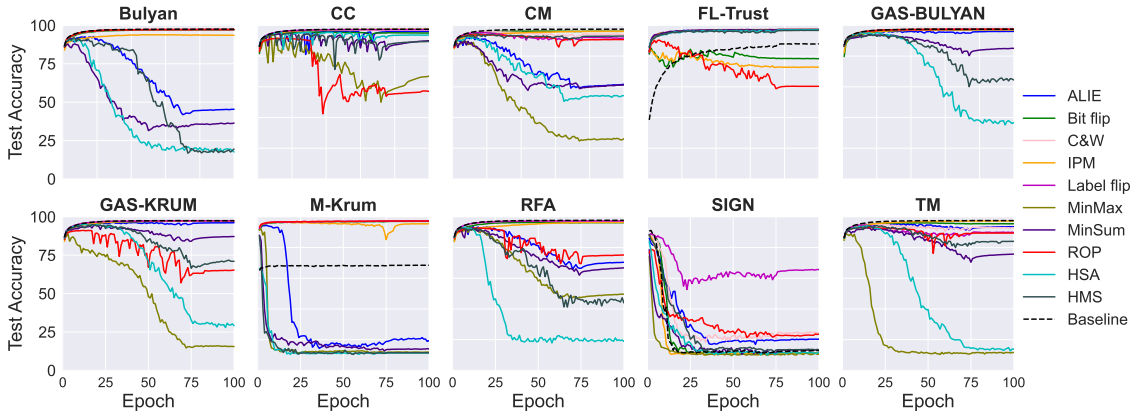


(a)

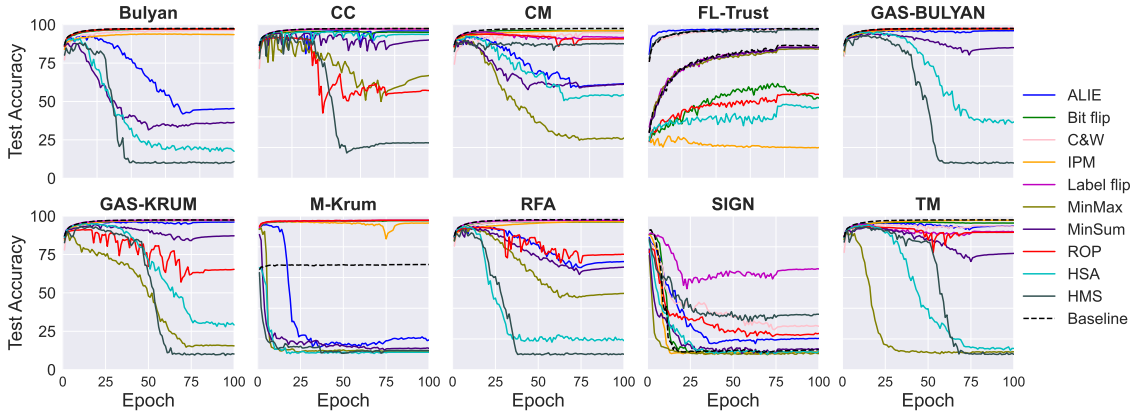


(b)

Figure 6.12: Test accuracy results of training 2-Layer CNN architecture with F-MNIST dataset distributed IID (6.12a) and non-IID(6.12b) over $k = 25$ clients $k_m = 5$ of them being malicious. Training and testing are performed over 100 epochs and repeated for 10 different aggregation mechanisms under 10 Byzantine attack strategies. The reported results are obtained by averaging 3 independent trials.



(a)



(b)

Figure 6.13: Test accuracy results of training 2-layer MLP architecture with MNIST dataset distributed IID (6.13b) and non-IID(6.13a) over $k = 25$ clients $k_m = 5$ of them being malicious. Training and testing are performed over 100 epochs and repeated for 10 different aggregation mechanisms under 10 Byzantine attack strategies. The reported results are obtained by averaging 3 independent trials.

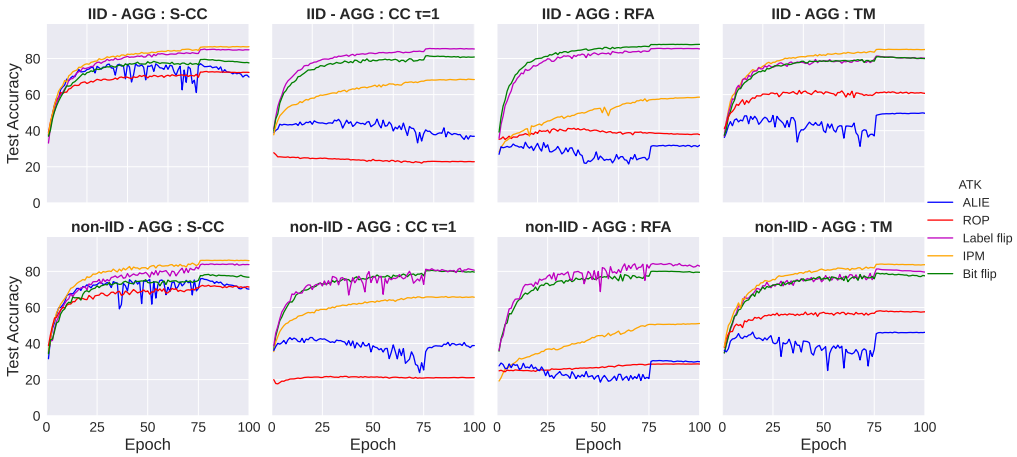


Figure 6.14: ResNet-20 trained on CIFAR-10 with β =0 on IID and non-IID across the clients. Test accuracy with various aggregation and attack methods in the presence of attacks.

considered here, while keeping the same computational complexity as the CC scheme. Thus we recommend employing the S-CC aggregator with local momentum $\beta = 0.9$, where it can achieve near baseline accuracy against all the considered attacks. In the next section, we will go over the Results for β=0.9

Finally in Fig. 6.15, we compare our proposed S-CC aggregator against the most successful robust aggregators that we considered. Overall S-CC provides better test accuracy and smoother convergences compared to the other aggregators, especially in non-IID simulations as seen in the second row of the 6.15. Unfortunately, S-CC can not fully defend against the MinMax attack while still being able to converge. This is mainly the result of the curse of dimensionality where MinMax fully exploits the norm distance of the perturbation while S-CC in its core is still a norm-based aggregor. Ultimately, S-CC can offer superior robustness compared to the other robust aggregators with similar complexity such as CC, RFA, and TM.

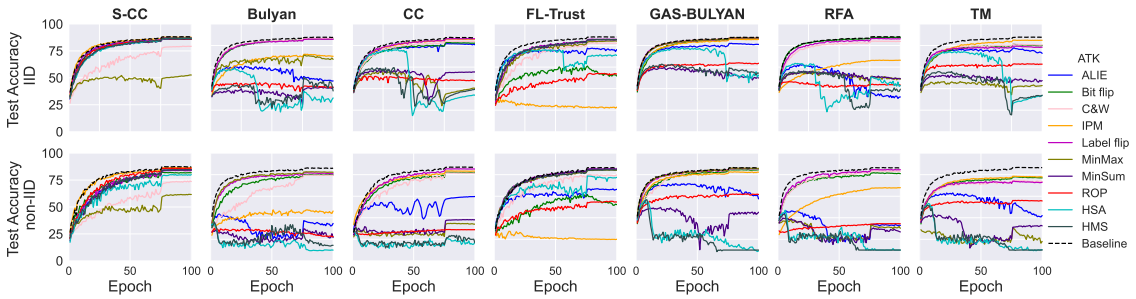


Figure 6.15: Test accuracy results of training ResNet-20 architecture with CIFAR-10 dataset distributed IID and non-IID over $k = 25$ clients $k_m = 5$ of them being malicious. Proposed S-CC (3.5) against the other most successful robust aggregators under 10 Byzantine attack strategies.

6.5 Runtime analysis

6.5.1 Attack times

In this section, we go over the runtime of the untargeted poisoning attacks in practical scenarios. We experimented with $k = [25, 50, 100]$ total clients and calculated the average attack generation time on ResNet-20 architecture which has a total of $\sim 7 \times 10^5$ trainable parameters ($d \cong 700K$). All simulations run on the NVIDIA A40 GPU and averaged over 1000 aggregations. For our non-omniscient attacks such as bit-flip or label-flip, we can achieve near-constant complexity if we do not include the gradient calculation. Furthermore, since this attack does not employ updates from benign clients, their runtime is also not affected with respect to the total number of clients. Omniscient attacks such as IPM, ALIE, and our proposed ROP and HSA, can achieve asymptotic complexity of $O(dk)$ with very similar run times. However, ROP employs multiple operations to angle manipulation to vector rejection, as such it takes slightly more run time than the IPM and ALIE. Optimized attacks, namely MinMax, MinSum, and our proposed HMS have asymptotic complexity of $O(dk^2)$ due to the optimization process. Their actual runtime is considerably slower than the non-optimized omniscient attacks while HMS can able to run faster at a large number of clients, particularly on $k = 100$. This is mainly due to the reduced threshold compared to the MinSum, as such it optimization process on HMS is

relatively faster.

6.5.2 Aggregation times

In this section, we go over the runtime of the robust aggregator in practical scenarios. We experimented with $k = [25, 50, 100]$ total clients and calculated the average aggregation time on ResNet-20 architecture which has a total of $\sim 7 \times 10^5$ trainable parameters ($d \cong 700K$). All simulations run on the NVIDIA A40 GPU and averaged over 1000 aggregations. We show those results in Table 6.7. Out of all aggregators, FedAvg [McMahan et al., 2017], SignSGD [Bernstein et al., 2019] CM, and TM [Yin et al., 2018] are considerably faster than all aggregators with asymptotic runtime of $O(dk)$. Following that, CC [Karimireddy et al., 2021] and RFA and FL-Trust [Cao et al., 2021b] which also theoretically have asymptotic runtime of $O(dk)$. Finally, as expected, M-Krum and Bulyan are the slowest robust aggregators with an asymptotic runtime of $O(dk^2)$. Our simulations for the robust aggregators also show the expected linear/polynomial increase in the runtime as the k increases. Theoretically, [Liu et al., 2023] should achieve the same performance as the aggregator that is employed. However, due to the sub-optimal implementations, GAS with $p = 1000$ considerably slows down the M-Krum and Bulyan. Finally, our proposed S-CC aggregator also achieved the asymptotic runtime of $O(dk)$ with near similar performance to the CC aggregator. We also note that S-CC employs cosine-similarity based grouping then a bucketing approach as well as an iterative reference update mechanism resulting in higher computation time compared to the CC.

Table 6.1: Hyper-parameter search on attack location ρ, reference point λ, and angle of the perturbation π. Reporting the test accuracy for the CIFAR-10 image classification task. The lowest score for each aggregator for the respective simulation setup is denoted in **bold**

	IID $\beta = 0.9$			IID $\beta = 0.99$			non-IID $\beta = 0.9$		
Atk(.) setup	CC	TM	RFA	CC	TM	RFA	CC	TM	RFA
$\rho = 0, \lambda = 0, \pi = 60$	74.62 ± 1.76	74.55 ± 2.34	74.3 ± 0.55	79.97 ± 0.39	79.48 ± 1.18	81.71 ± 0.38	65.54 ± 2.6	72.31 ± 0.46	66.75 ± 1.51
$\rho = 0, \lambda = 0, \pi = 90$	69.4 ± 1.76	70.58 ± 0.34	65.51 ± 2.02	71.34 ± 0.37	73.04 ± 0.36	74.51 ± 0.27	57.44 ± 1.4	62.53 ± 0.28	54.14 ± 2.06
$\rho = 0, \lambda = 0, \pi = 120$	65.42 ± 1.84	69.47 ± 0.92	54.4 ± 0.88	64.56 ± 0.05	62.59 ± 4.93	64.54 ± 2.28	48.9 ± 0.54	57.59 ± 2.26	39.06 ± 1.27
$\rho = 0, \lambda = 0, \pi = 135$	66.52 ± 0.06	65.62 ± 0.5	50.73 ± 0.12	61.72 ± 2.34	60.93 ± 1.77	55.19 ± 1.42	50.02 ± 1.18	60.0 ± 1.85	32.98 ± 1.61
$\rho = 0, \lambda = 0, \pi = 180$	84.5 ± 0.28	84.86 ± 0.18	65.84 ± 0.14	77.79 ± 0.06	76.18 ± 0.74	74.53 ± 0.16	82.94 ± 0.06	79.33 ± 0.7	60.93 ± 0.75
$\rho = 0, \lambda = 0.5, \pi = 60$	73.1 ± 1.46	69.76 ± 1.63	66.6 ± 1.19	68.64 ± 0.64	74.32 ± 0.48	74.21 ± 0.13	56.45 ± 0.8	64.42 ± 0.75	41.88 ± 1.89
$\rho = 0, \lambda = 0.5, \pi = 90$	66.74 ± 1.14	69.88 ± 1.32	50.83 ± 1.74	60.94 ± 2.08	67.07 ± 1.66	56.4 ± 0.97	48.08 ± 0.18	57.26 ± 0.43	34.18 ± 0.7
$\rho = 0, \lambda = 0.5, \pi = 120$	66.86 ± 0.64	69.27 ± 0.57	37.09 ± 0.73	59.77 ± 1.48	67.43 ± 1.41	**44.65 ± 1.38**	48.96 ± 0.62	59.13 ± 1.06	25.15 ± 2.38
$\rho = 0, \lambda = 0.5, \pi = 135$	66.64 ± 0.72	70.8 ± 0.01	**30.22 ± 3.31**	60.33 ± 1.38	63.04 ± 4.03	50.1 ± 1.69	47.58 ± 1.01	59.89 ± 1.24	**22.63 ± 1.98**
$\rho = 0, \lambda = 0.5, \pi = 180$	85.11 ± 0.25	83.96 ± 1.0	62.4 ± 0.86	75.61 ± 0.81	75.6 ± 0.4	68.08 ± 0.94	81.81 ± 0.71	78.95 ± 0.94	48.79 ± 2.98
$\rho = 0, \lambda = 1, \pi = 60$	70.76 ± 0.64	69.98 ± 0.62	57.49 ± 0.5	60.09 ± 0.32	68.32 ± 1.47	58.74 ± 2.35	42.62 ± 2.6	64.78 ± 0.43	40.7 ± 1.8
$\rho = 0, \lambda = 1, \pi = 90$	63.25 ± 1.15	67.22 ± 1.9	58.24 ± 0.02	63.34 ± 1.15	68.53 ± 2.8	63.96 ± 2.37	46.2 ± 1.1	59.63 ± 1.78	43.94 ± 1.02
$\rho = 0, \lambda = 1, \pi = 120$	67.46 ± 1.54	72.51 ± 0.31	60.29 ± 0.72	67.58 ± 0.63	70.99 ± 0.33	70.51 ± 0.93	55.86 ± 0.3	65.7 ± 2.44	53.0 ± 1.89
$\rho = 0, \lambda = 1, \pi = 135$	68.72 ± 0.8	75.48 ± 0.53	66.25 ± 0.62	73.2 ± 0.78	73.39 ± 0.21	73.1 ± 0.43	62.06 ± 0.72	68.86 ± 1.31	59.02 ± 2.18
$\rho = 0, \lambda = 1, \pi = 180$	83.36 ± 0.62	74.18 ± 0.57	85.24 ± 0.04	84.17 ± 0.36	83.92 ± 0.09	84.38 ± 0.16	74.47 ± 4.82	80.69 ± 0.77	83.65 ± 0.35
$\rho = 0.5, \lambda = 0, \pi = 45$	79.2 ± 1.74	78.75 ± 0.63	77.94 ± 0.08	82.5 ± 0.48	83.84 ± 0.05	83.66 ± 0.47	64.54 ± 0.3	76.36 ± 1.33	71.34 ± 1.68
$\rho = 0.5, \lambda = 0, \pi = 60$	73.91 ± 0.96	73.27 ± 0.25	73.48 ± 0.96	78.79 ± 1.12	79.94 ± 0.56	81.19 ± 0.24	62.55 ± 0.02	67.66 ± 1.39	65.64 ± 1.08
$\rho = 0.5, \lambda = 0, \pi = 90$	67.13 ± 0.24	66.51 ± 0.7	64.76 ± 1.29	67.55 ± 2.8	71.62 ± 0.8	75.3 ± 0.35	50.4 ± 1.56	57.47 ± 2.15	50.4 ± 1.71
$\rho = 0.5, \lambda = 0, \pi = 120$	62.43 ± 0.78	63.54 ± 1.74	56.21 ± 0.96	57.65 ± 0.59	64.9 ± 1.31	64.27 ± 1.86	45.53 ± 2.59	52.5 ± 3.33	38.24 ± 2.75
$\rho = 0.5, \lambda = 0, \pi = 135$	64.72 ± 0.31	63.4 ± 1.17	55.61 ± 2.06	56.16 ± 1.57	**58.85 ± 0.22**	59.64 ± 0.28	42.62 ± 1.86	55.41 ± 0.38	34.75 ± 0.48
$\rho = 0.5, \lambda = 0, \pi = 180$	84.38 ± 0.14	83.76 ± 0.3	75.84 ± 0.12	78.22 ± 0.42	76.28 ± 0.47	77.12 ± 1.06	82.35 ± 0.36	75.47 ± 1.1	67.69 ± 0.08
$\rho = 0.5, \lambda = 1, \pi = 60$	64.32 ± 0.1	66.16 ± 2.16	51.94 ± 0.86	56.32 ± 1.52	67.78 ± 1.26	59.39 ± 1.46	36.55 ± 3.87	57.72 ± 2.99	39.61 ± 0.39
$\rho = 0.5, \lambda = 1, \pi = 90$	59.14 ± 1.2	65.63 ± 1.44	54.02 ± 0.74	59.32 ± 2.04	69.46 ± 1.3	61.13 ± 1.53	38.74 ± 0.5	57.86 ± 1.37	40.1 ± 1.72
$\rho = 0.5, \lambda = 1, \pi = 120$	60.86 ± 0.76	71.09 ± 0.96	57.49 ± 0.21	65.48 ± 2.62	70.2 ± 0.34	67.14 ± 1.0	49.18 ± 0.4	59.7 ± 1.72	43.98 ± 2.4
$\rho = 0.5, \lambda = 1, \pi = 135$	65.24 ± 1.14	73.39 ± 1.24	59.14 ± 1.06	70.03 ± 0.68	70.26 ± 0.22	68.51 ± 2.32	56.76 ± 0.2	62.95 ± 3.02	44.72 ± 0.72
$\rho = 0.5, \lambda = 1, \pi = 180$	81.13 ± 0.08	73.08 ± 0.08	84.24 ± 0.14	82.57 ± 0.5	83.18 ± 0.72	84.35 ± 0.08	75.31 ± 0.48	79.23 ± 0.97	82.08 ± 1.04
$\rho = 1, \lambda = 0, \pi = 60$	72.95 ± 0.81	71.41 ± 0.16	77.84 ± 1.2	56.32 ± 1.52	79.7 ± 0.1	81.78 ± 0.35	53.87 ± 1.48	64.83 ± 2.02	61.99 ± 2.75
$\rho = 1, \lambda = 0, \pi = 90$	66.06 ± 1.54	63.78 ± 0.45	66.27 ± 1.93	66.31 ± 0.06	68.94 ± 0.22	73.9 ± 1.06	42.37 ± 3.8	55.06 ± 0.52	48.34 ± 2.59
$\rho = 1, \lambda = 0, \pi = 120$	61.7 ± 1.29	**59.42 ± 1.96**	59.28 ± 0.42	51.92 ± 0.0	66.7 ± 0.14	65.43 ± 0.37	40.41 ± 1.84	49.21 ± 1.06	41.15 ± 1.27
$\rho = 1, \lambda = 0, \pi = 135$	59.11 ± 1.46	60.09 ± 1.01	57.63 ± 1.33	48.44 ± 1.36	65.54 ± 0.3	64.59 ± 1.68	36.44 ± 0.9	**47.74 ± 2.82**	38.73 ± 0.94
$\rho = 1, \lambda = 0, \pi = 180$	79.66 ± 0.4	83.56 ± 0.18	78.46 ± 0.04	77.6 ± 0.18	73.66 ± 0.58	78.3 ± 0.3	82.09 ± 0.19	74.09 ± 1.24	82.31 ± 1.53
$\rho = 1, \lambda = 0.5, \pi = 60$	67.86 ± 0.12	65.67 ± 1.1	68.53 ± 0.56	58.76 ± 0.3	71.89 ± 1.6	74.79 ± 0.69	29.7 ± 0.18	56.87 ± 1.22	42.81 ± 0.67
$\rho = 1, \lambda = 0.5, \pi = 90$	59.88 ± 1.1	61.26 ± 1.25	55.5 ± 0.14	50.5 ± 0.52	68.51 ± 0.39	60.48 ± 4.17	37.16 ± 0.38	54.25 ± 1.2	34.75 ± 0.88
$\rho = 1, \lambda = 0.5, \pi = 120$	56.36 ± 1.03	62.98 ± 0.46	45.28 ± 3.65	50.24 ± 2.12	60.91 ± 1.44	56.32 ± 0.98	32.55 ± 1.17	49.54 ± 3.02	33.49 ± 1.06
$\rho = 1, \lambda = 0.5, \pi = 135$	55.2 ± 1.68	65.5 ± 1.75	47.92 ± 1.01	52.74 ± 0.86	65.12 ± 0.1	54.17 ± 3.76	33.95 ± 3.45	52.47 ± 2.49	34.41 ± 2.39
$\rho = 1, \lambda = 0.5, \pi = 180$	77.22 ± 0.04	83.62 ± 0.44	74.9 ± 0.57	74.42 ± 1.1	73.74 ± 0.66	73.22 ± 0.1	76.93 ± 0.64	75.72 ± 0.57	79.61 ± 1.69
$\rho = 1, \lambda = 0.9, \pi = 60$	60.9 ± 0.58	64.58 ± 0.98	59.2 ± 2.06	50.93 ± 0.62	68.66 ± 0.8	56.73 ± 1.95	28.98 ± 2.57	55.85 ± 1.37	47.86 ± 1.0
$\rho = 1, \lambda = 0.9, \pi = 90$	46.23 ± 1.03	64.12 ± 0.54	34.26 ± 0.42	**49.32 ± 0.4**	63.82 ± 2.46	56.49 ± 1.3	**28.86 ± 0.26**	55.55 ± 1.81	34.26 ± 0.42
$\rho = 1, \lambda = 0.9, \pi = 120$	40.42 ± 1.96	66.54 ± 0.48	47.15 ± 2.6	57.18 ± 0.14	69.9 ± 0.2	60.5 ± 0.84	27.26 ± 2.84	56.03 ± 2.02	32.46 ± 1.07
$\rho = 1, \lambda = 0.9, \pi = 135$	44.78 ± 1.2	66.49 ± 1.57	47.51 ± 0.33	61.53 ± 1.36	70.78 ± 1.22	62.04 ± 0.76	31.08 ± 1.56	58.76 ± 1.24	33.16 ± 1.73
$\rho = 1, \lambda = 0.9, \pi = 180$	68.32 ± 0.04	83.57 ± 0.67	71.32 ± 0.2	78.47 ± 0.01	82.04 ± 1.06	79.68 ± 0.24	63.46 ± 1.36	75.76 ± 1.79	63.91 ± 1.91
$\rho = 1, \lambda = 1, \pi = 60$	58.69 ± 1.18	64.78 ± 0.62	44.98 ± 6.9	53.72 ± 0.6	67.88 ± 0.06	58.6 ± 2.19	29.7 ± 1.34	56.57 ± 0.95	35.56 ± 0.64
$\rho = 1, \lambda = 1, \pi = 90$	**40.08 ± 4.93**	62.5 ± 0.06	48.2 ± 4.14	57.02 ± 1.76	67.1 ± 0.6	60.3 ± 0.45	35.77 ± 1.67	55.22 ± 0.86	37.85 ± 0.73
$\rho = 1, \lambda = 1, \pi = 120$	51.84 ± 1.03	67.36 ± 0.42	52.54 ± 0.32	63.75 ± 0.14	69.73 ± 0.11	61.04 ± 1.14	39.39 ± 2.1	57.47 ± 1.68	36.05 ± 1.6
$\rho = 1, \lambda = 1, \pi = 135$	57.11 ± 2.46	69.75 ± 0.2	53.56 ± 0.42	68.96 ± 1.66	69.62 ± 2.43	64.91 ± 0.71	47.06 ± 0.32	61.63 ± 2.86	36.35 ± 0.91
$\rho = 1, \lambda = 1, \pi = 180$	76.18 ± 1.58	84.67 ± 0.68	79.09 ± 0.22	81.12 ± 0.9	82.35 ± 0.37	81.98 ± 0.46	71.3 ± 1.21	77.19 ± 1.41	68.29 ± 1.38

Table 6.2: Cifar-10 IID test accuracy results on different numbers of Byzantines with a total of 25 clients. Lower is better.

k_m		β=0				β=0.9				β=0.99			
	Attack / Aggr	**cc (τ =0.1)**	**cc**	**rfa**	**tm**	**cc (τ =0.1)**	**cc**	**rfa**	**tm**	**cc (τ =0.1)**	**cc**	**rfa**	**tm**
1	ROP	**56.6**	**60.4**	81.2	83.2	**79.98**	**85.40**	**83.27**	**81.98**	**80.95**	**85.10**	**85.65**	85.86
	Alie	86.9	87.0	87.4	87.3	82.88	87.73	87.49	87.74	82.20	86.74	86.26	86.07
	IPM	84.92	85.82	87.00	86.26	82.15	87.79	87.20	87.01	81.54	86.39	86.06	85.37
	Bit-Flip	86.26	86.82	87.09	86.39	80.93	87.30	86.44	86.97	82.03	85.85	86.04	**85.16**
	Label-flip	86.30	87.00	86.87	85.60	82.20	87.77	87.10	86.58	82.02	86.63	86.48	85.77
2	ROP	**40.77**	**45.05**	**72.71**	**71.05**	**71.29**	**79.96**	**79.66**	**63.93**	**79.63**	**79.55**	**81.86**	**82.39**
	Alie	80.73	79.74	67.32	74.98	82.94	87.47	84.71	85.84	82.29	86.75	85.71	85.70
	IPM	82.26	84.21	85.85	85.29	82.28	86.67	85.65	86.09	80.72	86.76	84.02	84.61
	Bit-Flip	84.94	85.48	85.99	85.12	82.31	86.57	86.36	85.59	80.44	85.20	85.46	84.25
	Label-flip	86.09	86.76	86.81	84.94	82.66	86.99	86.77	85.19	81.04	86.15	86.37	85.10
3	ROP	**31.91**	34.44	60.84	60.64	**62.67**	**71.57**	66.49	**65.76**	**75.63**	**65.88**	**70.91**	**76.13**
	Alie	34.16	**29.42**	**50.73**	**53.83**	80.94	85.81	**60.04**	73.38	83.30	85.86	84.09	83.65
	IPM	78.39	81.42	83.86	84.41	80.94	86.46	82.39	85.32	79.18	85.28	80.22	83.59
	Bit-Flip	84.63	84.55	84.63	84.34	80.32	85.21	85.08	84.79	79.56	83.64	84.01	83.69
	Label-flip	84.73	86.52	86.32	83.42	82.17	86.65	86.94	83.40	80.86	86.40	86.05	84.47
7	ROP	**10.52**	**22.61**	**24.74**	**22.09**	**28.36**	**33.95**	37.60	**38.51**	**44.47**	**45.82**	**46.59**	**49.23**
	Alie	33.21	32.52	25.99	22.16	59.97	54.80	**33.50**	45.07	71.38	79.97	56.53	52.93
	IPM	45.28	58.24	36.35	50.87	61.79	84.73	54.58	68.70	60.85	84.91	62.01	70.71
	Bit-Flip	74.04	73.17	74.75	75.07	76.13	77.40	75.60	76.02	74.94	78.15	78.14	77.97
	Label-flip	82.92	83.33	83.55	77.09	79.12	86.01	85.03	74.37	78.00	84.87	84.97	78.62
8	ROP	**10.62**	**19.83**	22.69	**11.37**	**23.39**	**33.35**	**35.43**	**33.86**	**32.62**	**40.28**	**43.92**	52.83
	Alie	29.19	31.39	**21.09**	18.36	51.77	27.37	48.70	39.70	63.08	75.23	56.09	**48.79**
	IPM	22.23	34.87	28.59	31.75	55.61	83.71	50.33	62.68	64.21	84.05	59.81	66.92
	Bit-Flip	69.47	69.71	70.52	69.65	73.82	74.13	71.81	72.19	74.62	74.53	76.12	75.38
	Label-flip	80.85	81.78	81.87	72.74	78.65	85.29	84.39	71.64	76.32	84.99	84.46	76.51
10	ROP	**10.66**	19.53	13.99	**12.39**	**19.88**	**33.52**	**32.74**	**21.14**	37.25	**32.05**	**40.00**	40.83
	Alie	19.98	19.25	16.41	14.57	37.50	38.10	33.98	26.29	**34.80**	56.67	45.88	**39.11**
	IPM	14.56	**12.98**	**10.27**	12.40	21.95	81.67	41.62	48.86	59.04	83.09	54.02	57.30
	Bit-Flip	56.77	62.73	59.86	51.43	63.02	62.79	63.42	58.90	65.01	67.12	65.93	66.23
	Label-flip	75.23	74.92	76.56	72.48	75.39	82.47	82.06	67.90	73.52	83.79	84.29	72.50
12	ROP	10.00	10.01	10.01	**10.63**	**10.36**	31.40	27.13	**17.96**	**22.51**	**25.04**	**33.84**	34.32
	Alie	16.92	16.79	13.87	16.49	28.64	**24.78**	**24.00**	20.80	31.17	40.00	36.64	**28.92**
	IPM	**7.68**	**8.84**	**10.00**	10.97	10.80	75.21	24.38	31.94	35.79	82.38	34.10	47.08
	Bit-Flip	33.91	33.90	36.71	36.71	33.02	38.27	40.62	38.94	46.24	48.24	47.20	49.53
	Label-flip	53.85	52.07	51.04	59.58	60.89	57.37	58.96	61.49	62.67	69.97	69.77	62.23

Table 6.3: CIFAR-10 test accuracy results on different numbers of Byzantines with a total of 25 clients on the non-IID dataset. Lower is better.

k_m		β=0				β=0.9				β=0.99			
	Attack / Aggr	**cc**	**cc**	**rfa**	**tm**	**cc**	**cc**	**rfa**	**tm**	**cc**	**cc**	**rfa**	**tm**
1	ROP	**55.60**	**64.59**	**44.61**	**57.53**	**66.06**	**81.50**	**77.56**	**72.30**	**47.42**	**75.24**	***68.88***	**72.51**
	Alie	84.85	85.97	87.04	86.09	77.15	86.49	85.99	81.30	52.27	84.64	84.46	76.63
	IPM	83.18	84.99	86.92	85.79	77.04	86.52	84.89	78.95	51.09	84.36	81.97	73.34
	Bit-Flip	84.92	85.31	86.65	85.14	76.05	86.05	85.99	80.27	46.83	84.00	84.58	73.96
	Label-flip	84.67	84.71	86.58	85.12	75.39	86.23	85.98	78.98	48.86	84.77	84.17	**71.04**
2	ROP	**38.02**	**53.23**	**30.54**	**39.52**	**50.07**	**63.03**	**52.83**	**56.02**	**28.94**	**51.07**	**50.68**	**61.42**
	Alie	73.80	75.72	56.03	74.30	75.29	85.34	75.96	78.31	56.68	84.02	75.19	70.48
	IPM	79.87	82.76	85.72	84.55	77.42	86.13	80.50	77.76	49.02	84.02	76.87	70.92
	Bit-Flip	84.92	85.31	86.65	85.14	75.20	85.20	85.01	76.72	48.28	83.52	83.29	71.82
	Label-flip	84.67	84.71	86.58	85.12	75.49	86.15	86.51	77.33	48.49	84.12	84.13	71.19
3	ROP	**27.57**	30.28	**21.86**	**32.32**	**28.91**	**43.76**	43.36	51.36	**17.11**	**36.95**	44.01	**49.26**
	Alie	41.35	**27.48**	46.79	47.46	67.65	83.66	**34.82**	**43.86**	49.62	80.99	**41.75**	63.46
	IPM	76.27	78.12	83.21	83.41	76.19	86.10	74.26	75.67	51.24	83.63	73.28	67.77
	Bit-Flip	82.17	82.34	84.27	82.59	75.98	84.14	83.36	77.22	45.99	82.63	82.23	71.16
	Label-flip	81.71	84.54	85.76	82.73	74.04	85.98	84.85	74.44	46.39	83.93	84.37	71.07
7	ROP	**11.27**	**11.26**	**14.19**	**16.19**	**14.87**	**24.78**	23.90	**18.87**	**12.43**	**22.61**	**21.96**	20.90
	Alie	26.82	27.80	20.63	16.88	34.01	35.23	**22.19**	22.58	20.21	31.13	23.37	**19.04**
	IPM	44.88	55.30	44.63	43.44	58.66	82.04	71.80	48.78	31.76	81.54	68.87	47.04
	Bit-Flip	74.76	74.64	73.84	69.36	67.26	77.84	78.26	66.95	45.14	76.03	76.99	53.81
	Label-flip	81.41	73.21	80.42	76.58	66.01	83.24	83.72	69.70	37.70	80.81	82.62	60.47
8	ROP	**12.00**	**11.56**	14.58	**12.52**	**12.72**	**24.36**	**18.72**	**16.19**	**10.39**	**21.80**	**21.12**	22.26
	Alie	14.48	18.97	**13.51**	14.33	31.97	26.65	23.01	19.90	18.58	22.03	21.30	**18.37**
	IPM	10.00	9.93	10.84	10.91	55.24	64.23	63.57	42.29	26.17	79.85	65.34	38.65
	Bit-Flip	51.02	36.79	57.90	49.13	64.21	76.23	75.02	65.78	44.89	73.30	74.03	52.31
	Label-flip	58.22	61.00	64.00	68.24	69.46	83.01	82.37	64.23	36.91	80.41	81.86	60.77
10	ROP	**10.00**	**10.00**	**10.10**	**9.94**	**11.13**	**17.19**	**10.00**	**13.23**	**10.79**	18.38	**11.04**	14.79
	Alie	15.52	16.20	14.35	16.82	18.66	18.01	14.41	16.81	19.33	**14.27**	20.76	**12.81**
	IPM	**10.00**	**10.00**	**10.00**	10.92	31.18	36.00	52.30	26.45	17.48	74.74	66.48	25.79
	Bit-Flip	12.80	20.42	10.00	14.47	39.45	60.63	54.30	44.80	32.99	60.90	52.81	36.19
	Label-flip	29.11	51.51	44.65	47.81	56.39	75.56	79.15	60.50	37.51	69.52	80.12	54.04
12	ROP	10.00	10.01	10.01	**10.63**	10.00	15.53	**10.00**	**10.01**	**10.00**	19.40	**10.00**	**10.80**
	Alie	16.92	16.79	13.87	16.49	17.10	**13.78**	15.47	15.55	14.30	14.49	17.91	17.93
	IPM	**7.68**	**8.84**	**10.00**	10.97	**9.90**	12.64	**10.00**	15.76	10.73	**10.11**	10.34	14.37
	Bit-Flip	33.91	33.90	36.71	36.71	11.42	10.98	12.48	20.29	17.16	20.66	10.54	16.84
	Label-flip	53.85	52.07	51.04	59.58	44.44	61.10	44.07	43.17	26.41	41.58	62.52	38.05

Table 6.4: Final test accuracy results of training ResNet-20 architecture with Cifar-10 dataset distributed IID over $k = 25$ client $k_m = 5$ of them being malicious. The training is performed over 100 epochs and repeated for 8 different aggregation mechanisms and under 8 different Byzantine attack strategies. The reported results are obtained by averaging 3 number of independent trials. We use **blue**, **red** and **green** to highlight the best results for fixed policy attacks, ALIE, Bit Flip, IPM, Label Flip, ROP dynamically optimized attacks, Min-Sum, Min-Max and proposed sparse hybrid attack variations with different sparsity constraint δ, respectively, and we underline the best attack result against each defence mechanism. For our proposed hybrid sparse attack variations, we consider 5 different sparsity levels; $\delta_1 = 5 \times 10^{-3}$, $\delta_2 = 1 \times 10^{-2}$, $\delta_3 = 5 \times 10^{-2}$, $\delta_4 = 2 \times 10^{-1}$, and $\delta_5 = 5 \times 10^{-1}$. Finally, we set $\delta_{FC}^{max} = 2.5 \times 10^{-1}$.

Method	**Bulyan**	**CC**	**CM**	**M-Krum**	**RFA**	**TM**	**GAS** Krum-Bulyan
No ATK	87.6 ± 1	87.4 ± 0.5	88 ± 0.1	89.2 ± 0.7	88.2 ± 0.3	87.8 ± 0.4	87.7 ± 0.3 – 87.6 ± 0.4
ALIE	47.1 ± 16	80.7 ± 1.4	**45.1 ± 9.3**	**55.8 ± 14**	***32.3 ± 16***	72.8 ± 2.5	81 ± 0.9 – 81.1 ± 0.6
Bit Flip	86.2 ± 0.6	82.2 ± 0.7	80.9 ± 0.8	87.8 ± 0.3	87.5 ± 0.7	80.1 ± 0.9	86.5 ± 0.5 – 86.8 ± 0.8
IPM	70 ± 0.9	85.8 ± 0.3	54.5 ± 0.8	83.8 ± 0.1	66.4 ± 0.7	84.9 ± 0.1	85.3 ± 0.2 – 85.6 ± 0.3
Label Flip	85.8 ± 0.3	86.4 ± 0.6	79.9 ± 0.1	87.7 ± 0.1	86.3 ± 0.1	78.5 ± 0.3	87.1 ± 0.7 – 86.9 ± 0.6
ROP	**41.5 ± 2**	**47.4 ± 0.9**	79 ± 0.4	88.2 ± 0.5	43.2 ± 2.2	**62.7 ± 1.3**	**612 ± 1.6 – 63.5 ± 0.7**
Best Attack *	41.5 ± 2	47.4 ± 0.9	45.1 ± 9.3	55.8 ± 14	32.3 ± 16	62.7 ± 1.3	62 ± 1.6 – 63.5 ± 0.7
Min-Sum	41.1 ± 0.5	55.5 ± 5.2	**47.6 ± 0.4**	55.1 ± 1.3	49.4 ± 8.6	47.5 ± 7.4	55.1 ± 3 – **49.9 ± 11.8**
Min-Max	67.2 ± 4.4	40.3 ± 7.8	48 ± 1.8	61.6 ± 12	48.7 ± 2.8	42.9 ± 6.2	**47.1 ± 4** – 86.6 ± 0.2
HMS	**37.9 ± 5.6**	**39.3 ± 3**	47.7 ± 1	**14.5 ± 1.4**	**39.5 ± 0.9**	**33.9 ± 5.2**	47.2 ± 10 – 54.4 ± 1.2
$\Pi_r(\delta_1)$	43 ± 18	79.3 ± 0.4	40 ± 16	54.9 ± 5.3	42.7 ± 6.3	70.6 ± 1.3	78.9 ± 1 – 78.5 ± 1.3
$\Pi_r(\delta_3)$	47.6 ± 3.1	75.1 ± 0.8	45.9 ± 5.9	14.1 ± 2	48.4 ± 11	70.2 ± 1.9	75.8 ± 0.6 – 75.7 ± 1.3
$\Pi_r(\delta_4)$	40.2 ± 0.5	41.6 ± 2.5	49.3 ± 8.5	12.1 ± 0.7	59.4 ± 0.6	61.9 ± 2.4	65 ± 1.3 – 66.3 ± 2.2
$\Pi_r(\delta_5)$	28.2 ± 1.9	46.8 ± 2.8	42.1 ± 6.5	***11.4 ± 1.2***	47.8 ± 5.9	40 ± 5.5	48.3 ± 0.8 – 86.8 ± 0.5
$\Pi_r^+(\delta_1)$	37.9 ± 10	48.9 ± 21	45.5 ± 9.7	30.1 ± 12	45.2 ± 5.6	63.7 ± 12	80.5 ± 0.3 – 79 ± 1.2
$\Pi_r^+(\delta_3)$	37.5 ± 5.3	50.1 ± 9.8	49.6 ± 9.4	14.2 ± 0.7	50 ± 1.9	61.4 ± 5.9	72.2 ± 1.1 – 73.1 ± 1.1
$\Pi_r^+(\delta_4)$	26.8 ± 1.2	43 ± 1.3	45.5 ± 5.2	14.8 ± 1.4	39.3 ± 10	46.2 ± 2.4	61.5 ± 6 – 60.7 ± 5
$\Pi_r^+(\delta_5)$	27.6 ± 2.6	38.9 ± 0.6	43.1 ± 11	11.5 ± 0.9	**36.8 ± 2.9**	31.3 ± 9.7	***39.4 ± 7.3*** – 86.2 ± 0.1
$\Pi_{lr}(\delta_1)$	44.5 ± 17	79.2 ± 0.3	36.7 ± 1.8	54.7 ± 4.8	53 ± 3.8	71.7 ± 3.5	81.2 ± 1 – 80.2 ± 0.9
$\Pi_{lr}(\delta_3)$	53.1 ± 3.9	75.4 ± 1.9	36.3 ± 5.6	26.7 ± 11	53.9 ± 5.9	67.9 ± 2.9	74.2 ± 2.6 – 74.7 ± 0.7
$\Pi_{lr}(\delta_4)$	36.3 ± 2.5	41 ± 2.1	50.3 ± 5.2	13.3 ± 1.9	59.4 ± 2	62.1 ± 2.9	64.9 ± 3.2 – 61.9 ± 3.6
$\Pi_{lr}(\delta_5)$	26.3 ± 2.6	37.5 ± 8.2	50.6 ± 3.4	13.3 ± 3.6	48.4 ± 5.2	39.7 ± 1.4	48 ± 1.8 – 86.8 ± 0.5
$\Pi_{lr}^+(\delta_1)$	32.8 ± 4.4	74.7 ± 5.2	45.3 ± 6.3	13.8 ± 1	45.6 ± 4.3	56.6 ± 19	78.9 ± 1.2 – 77.3 ± 1.8
$\Pi_{lr}^+(\delta_3)$	32 ± 5.4	45.1 ± 2	35.9 ± 7.8	16.5 ± 2.9	56.8 ± 2.8	59 ± 7.1	73.2 ± 0.6 – 69.9 ± 1.9
$\Pi_{lr}^+(\delta_4)$	***26 ± 3***	34.3 ± 7.3	48 ± 1.8	17.7 ± 1.7	46.7 ± 4.8	42.3 ± 7.2	58.7 ± 6 – 56.1 ± 8.7
$\Pi_{lr}^+(\delta_5)$	26.3 ± 3.4	37.6 ± 7	45.3 ± 3.5	13.5 ± 1.9	42.7 ± 2.8	***30.6 ± 0.5***	39.2 ± 2.5 – 87 ± 0.4
$\Pi_{ERK}^+(\delta_1)$	26.7 ± 1.1	41 ± 5.2	44.4 ± 14	16.3 ± 1.3	45.3 ± 13	72.1 ± 0.6	78 ± 0.6 – 73.3 ± 1.6
$\Pi_{ERK}^+(\delta_3)$	57.4 ± 0.4	65.2 ± 9.3	47.1 ± 4.8	61.4 ± 7.8	69.9 ± 0.7	62.9 ± 11	70.8 ± 3.5 – 73.6 ± 1
$\Pi_{ERK}^+(\delta_4)$	58.5 ± 0.9	50.7 ± 11	55.4 ± 1.7	56.7 ± 6.8	65.6 ± 1.3	58.3 ± 4.5	61.1 ± 4.5 – 71.5 ± 2.1
$\Pi_{ERK}^+(\delta_5)$	85.3 ± 0.7	52.6 ± 10	50.1 ± 2.7	55 ± 15	59.7 ± 1.6	49.2 ± 3	51.9 ± 2.7 – 86.3 ± 0.2
$\Pi_F^+(\delta_1)$	40.6 ± 1.7	39.6 ± 3.6	39.5 ± 9.2	15.5 ± 1.2	52.8 ± 11	53.5 ± 17	74.5 ± 4 – 74.8 ± 3.3
$\Pi_F^+(\delta_1, \delta_{FC}^{max})$	31.3 ± 2.7	***33.9 ± 2.9***	***35.7 ± 4.5***	16.3 ± 2.1	42.7 ± 10	33.4 ± 3.3	59.5 ± 18 – **52.5 ± 8.3**
$\Pi_F^+(\delta_2, \delta_{FC}^{max})$	28.4 ± 1.9	38.7 ± 1	42.3 ± 18	16.3 ± 2.1	39.9 ± 9.4	47.5 ± 6.7	67.8 ± 8.2 – 59.3 ± 9.5

Table 6.5: Final test accuracy results of training ResNet-20 architecture with Cifar-10 dataset distributed non-IID over $k = 25$ client $k_m = 5$ of them being malicious. The training is performed over 100 epochs and repeated for 8 different aggregation mechanisms and under 8 different Byzantine attack strategies. The reported results are obtained by averaging 3 number of independent trials. We use **blue**, **red** and **green** to highlight the best results for fixed policy attacks, ALIE, Bit Flip, IPM, Label Flip, ROP dynamically optimized attacks, Min-Sum, Min-Max and proposed sparse hybrid attack variations with different sparsity constraint δ, respectively, and we underline the best attack result against each defence mechanism. For our proposed hybrid sparse attack variations, we consider 5 different sparsity levels; $\delta_1 = 5 \times 10^{-3}$, $\delta_2 = 1 \times 10^{-2}$, $\delta_3 = 5 \times 10^{-2}$, $\delta_4 = 2 \times 10^{-1}$, and $\delta_5 = 5 \times 10^{-1}$. Finally, we set $\delta_{FC}^{max} = 2.5 \times 10^{-1}$.

Method	**Bulyan**	**CC**	**CM**	**M-Krum**	**RFA**	**TM**	**GAS** Krum-Bulyan
No ATK	86 ± 0.5	86.9 ± 0.3	79.9 ± 0.4	88.1 ± 0.4	86.4 ± 0.7	86.7 ± 0.5	86.7 ± 0.5 – 86.3 ± 0.5
ALIE	34.7 ± 0.7	59.7 ± 3.4	34.5 ± 1.9	**49.7 ± 9.2**	**32.9 ± 3.4**	**42.1 ± 7.2**	**57 ± 11.4** – 63.6 ± 5.3
Bit Flip	79.8 ± 1	81.8 ± 0.5	73.5 ± 1.1	85.2 ± 0.5	80.9 ± 0.6	77 ± 1.3	84.2 ± 0.3 – 84.5 ± 0.2
IPM	46.3 ± 0.9	83.3 ± 0.7	61.9 ± 4.6	73.2 ± 3.9	67.9 ± 1.9	77.9 ± 1.9	83.3 ± 0.6 – 82.3 ± 0.7
Label Flip	80.9 ± 1	85.2 ± 0.4	72.4 ± 0.2	87.3 ± 0.3	84.6 ± 0	73 ± 1.5	86 ± 0.6 – 85.5 ± 0.3
ROP	**22.5 ± 5.4**	**28.9 ± 1.1**	**31.2 ± 2.4**	87.3 ± 0.7	34.4 ± 2	55.6 ± 1.6	64.6 ± 1.7 – **62.5 ± 1.2**
Best Attack *	22.5 ± 5.4	28.9 ± 1.1	31.2 ± 2.4	49.7 ± 9.2	34.4 ± 2	42.1 ± 7.2	57 ± 11.4 – 62.5 ± 1.2
Min-Sum	21.6 ± 4.1	38.1 ± 2.6	26.1 ± 2.5	27.2 ± 5.9	26.8 ± 2.2	32.5 ± 1.9	45.8 ± 4.7 – 45.1 ± 5.8
Min-Max	82.2 ± 0.1	34 ± 1.4	**26 ± 3**	86.9 ± 0.5	30.2 ± 2.2	18 ± 1.5	30.1 ± 16 – 85.9 ± 0.4
HMS	**14.4 ± 3.1**	**16.7 ± 1.5**	49 ± 13	**10 ± 0**	**10 ± 0**	**10 ± 0**	**10 ± 0** – **10 ± 0**
$\Pi_r(\delta_1)$	22.7 ± 2.4	47.4 ± 2.2	27 ± 1.7	27.4 ± 4.1	33.8 ± 10	36.6 ± 7.2	60 ± 5 – 48.1 ± 13
$\Pi_r(\delta_3)$	18.9 ± 2.2	25.2 ± 3.4	28.1 ± 1.7	13.6 ± 3.5	28.3 ± 6	37.8 ± 8.7	59.4 ± 3.2 – 43 ± 7.6
$\Pi_r(\delta_4)$	11.7 ± 1.4	19.6 ± 2.4	27.3 ± 1.1	10 ± 0.1	39.8 ± 6.3	23.9 ± 8.6	55.6 ± 8.8 – 23.4 ± 8
$\Pi_r(\delta_5)$	25.5 ± 4.2	***15.4 ± 3.5***	20.9 ± 2.9	9.9 ± 0.6	28.9 ± 8.9	***10 ± 0***	57.3 ± 5 – 53.3 ± 8.7
$\Pi_r^+(\delta_1)$	9.7 ± 0.6	20.1 ± 1.6	22.5 ± 4.2	11.9 ± 1.6	11.3 ± 1.8	34 ± 7.4	36.5 ± 11 – 31.1 ± 6.6
$\Pi_r^+(\delta_3)$	12.2 ± 2.2	17.3 ± 4.9	22.1 ± 1.4	12 ± 2.5	10 ± 0	25.7 ± 7.3	33.2 ± 13.9 – 10 ± 0
$\Pi_r^+(\delta_4)$	18 ± 6.5	18.3 ± 1.3	27.3 ± 1	11.1 ± 1.6	10 ± 0	10 ± 0	41.3 ± 6.2 – 15.6 ± 8.5
$\Pi_r^+(\delta_5)$	16.4 ± 2.4	16.6 ± 3.2	23.2 ± 1.1	9.7 ± 0.3	14.5 ± 3.3	***10 ± 0***	19.2 ± 3.4 – 60.4 ± 5
$\Pi_{lr}(\delta_1)$	24.1 ± 1.5	56.1 ± 1.8	32.8 ± 3	45.4 ± 6	28.9 ± 7.3	46.6 ± 1.4	45.3 ± 3.2 – 43.9 ± 11
$\Pi_{lr}(\delta_3)$	22.8 ± 4	25.7 ± 4	25.9 ± 5.2	9.4 ± 1.5	44.2 ± 7.5	43.2 ± 8.6	61.1 ± 4.6 – 57.2 ± 6.9
$\Pi_{lr}(\delta_4)$	17 ± 6.2	20.7 ± 2.5	27.6 ± 0.7	22 ± 7.6	28 ± 8.2	32.7 ± 1.6	51.1 ± 12.8 – 38 ± 5
$\Pi_{lr}(\delta_5)$	19.4 ± 4	15.8 ± 2	21 ± 4.1	11 ± 1.2	25.2 ± 2.5	***10 ± 0***	54.7 ± 4.5 – 48.2 ± 12
$\Pi_{lr}^+(\delta_1)$	10.1 ± 0	25.1 ± 1.1	28 ± 2.5	14 ± 2.1	10 ± 0	23.3 ± 5	34.7 ± 3.6 – 21.7 ± 4.2
$\Pi_{lr}^+(\delta_3)$	10.3 ± 1.3	16.7 ± 3.7	23.3 ± 3.5	10.3 ± 0.4	10 ± 0	27.4 ± 9.9	22.2 ± 9 – 11.3 ± 1.8
$\Pi_{lr}^+(\delta_4)$	12.6 ± 1.7	18.8 ± 0.4	24.7 ± 2.5	10.2 ± 0.3	12.1 ± 2.9	11.7 ± 1.2	35 ± 17 – 22.3 ± 13
$\Pi_{lr}^+(\delta_5)$	22.2 ± 4.2	17.8 ± 3.2	21.7 ± 1.4	9.7 ± 0.2	29 ± 7.8	***10 ± 0***	11.8 ± 2.4 – 51.6 ± 11
$\Pi_{ERK}^+(\delta_1)$	15.5 ± 6	15.8 ± 3.4	22.1 ± 1.4	10 ± 0.7	***9.4 ± 0.8***	10 ± 0	10 ± 0 – 10 ± 0
$\Pi_{ERK}^+(\delta_3)$	64 ± 8.4	61.6 ± 2.6	27.9 ± 3.6	48.9 ± 8.5	54.2 ± 4.7	41.3 ± 3.7	48 ± 10.8 – 71.4 ± 4.6
$\Pi_{ERK}^+(\delta_4)$	70 ± 3.2	52.7 ± 4.5	27.8 ± 2.1	46.8 ± 9.6	49.9 ± 4.4	26.2 ± 2.8	32.5 ± 2.2 – 71 ± 2.3
$\Pi_{ERK}^+(\delta_5)$	78.7 ± 3.5	41.6 ± 1.3	23.2 ± 2.4	47.9 ± 2.2	26.6 ± 6.5	20.7 ± 1.5	22.3 ± 3.5 – 72.3 ± 19
$\Pi_F^+(\delta_1)$	26.3 ± 3.4	24.3 ± 3	26.4 ± 2.3	10.9 ± 1.3	39.6 ± 5.8	10 ± 0	***10 ± 0*** – 64.3 ± 7.7
$\Pi_F^+(\delta_1, \delta_{FC}^{max})$	***9.9 ± 0.3***	17.6 ± 4	22.9 ± 2.1	9.8 ± 0.1	10 ± 0	***10 ± 0***	10 ± 0 – ***9.2 ± 1.1***
$\Pi_F^+(\delta_2, \delta_{FC}^{max})$	10.3 ± 0.4	16.2 ± 3.5	***19.9 ± 1.5***	***8.8 ± 2.3***	10 ± 0	***10 ± 0***	***10 ± 0*** – 10 ± 0

Table 6.6: Average attack generation times of the untargeted Byzantine attacks for total clients of $k = 25$, $k = 50$ and $k = 100$ reported in miliseconds (ms). Bit-flip and Label-flip attacks also include gradient calculation times.

ATK — k	$k = 25$	$k = 50$	$k = 100$	Coplexity
ALIE [Baruch et al., 2019]	0.47	0.91	2.64	$O(dk)$
Bit-Flip* [Li et al., 2019]	1.24 - 13.3	1.14 - 13.3	1.69 - 13.2	$O(d)$
IPM [Xie et al., 2020b]	0.43	0.91	2.06	$O(dk)$
Label-Flip* [Biggio et al., 2012]	0.05 - 12.5	0.06 - 12-7	0.06 - 12.7	$O(c)$
MinMax [Shejwalkar and Houmansadr, 2021]	31.75	63.97	213.78	$O(dk^2)$
MinSum [Shejwalkar and Houmansadr, 2021]	33.24	68.09	205.15	$O(dk^2)$
ROP (ours)	1.91	1.92	5.53	$O(dk)$
HSA (ours)	0.47	1.00	2.68	$O(dk)$
HMS (ours)	31.40	66.45	190.28	$O(dk^2)$

Table 6.7: Average aggregation times of the aggregators for total clients of $k = 25$, $k = 50$ and $k = 100$ reported in miliseconds (ms).

AGGR	$k = 25$	$k = 50$	$k = 100$
FedAVG [McMahan et al., 2017]	0.3	0.5	0.7
Bulyan [El Mhamdi et al., 2018]	50	225	1073
CC [Karimireddy et al., 2021]	2.3	4.5	8.9
CM [Yin et al., 2018]	0.8	1	2.6
FL-Trust [Cao et al., 2021b]	27	34	43
GAS-Bulyan [Liu et al., 2023]	239	323	333
GAS-Krum [Liu et al., 2023]	146	226	232
M-Krum [Blanchard et al., 2017]	49	228	956
RFA [Pillutla et al., 2022]	6.1	12.1	24.2
S-CC (Ours)	6.1	12.9	25.2
SignSGD [Bernstein et al., 2019]	0.3	0.5	0.7
TM [Yin et al., 2018]	0.9	1.6	3.5

Chapter 7

DISCUSSION AND CONCLUSIONS

The CC framework in [Karimireddy et al., 2021] proposed to utilize the acceleration technique of momentum SGD to increase the robustness of FL against Byzantine attacks. The advantage of local momentum is two-fold: First, it decreases the variance of the client updates, statistically reducing the available space for Byzantine attacks. Second, the consensus momentum from the previous iteration can be used to neutralize Byzantine attacks by taking it as a reference point and performing clipping accordingly. In this work, we showed that it is possible to circumvent the CC defence by redesigning existing attack mechanisms, such as ALIE and IPM, and the revised attacks can succeed against CC as well as other known defence mechanisms.

We highlighted two important aspects of the CC framework. First, it relies on the assumption that Byzantine attacks target benign updates. Hence, the CC mechanism considers the previous consensus update as the reference for clipping. We argue that CC benefits from the mismatch between the assumed target and the reference. Accordingly, it is possible to circumvent the CC defences by matching the target to the reference. Second, CC is an angle-invariant operation; that is the angle between the reference and the candidate vectors does not have an impact on the clipping operation. Based on these observations, we introduced a novel attack mechanism called ROP to circumvent the CC defences. We have shown through extensive numerical experiments that, ROP can successfully poison the model even when CC is deployed at PS as a defence mechanism. We have also shown that ROP is also effective against other well-known defence mechanisms, including TM and RFA as well. However, ROP does not utilize distances or variance among the client and uses fixed scaling hyper-parameter to scale the perturbation. While this approach quite effective in IID scenarios, due to the fixed pertubartion it limits the effectiveness of the ROP in non-IID scenarios. In future work, we will enhance the ROP with optimization methods such as MinMax and MinSum to employ distances among benign clients.

Overall, for the scope of this project, we publish those results in [Özfatura et al., 2024]

Later, we introduced a novel Byzantine attack framework that is simultaneously effective against different types of defence mechanisms. The proposed framework generates an attack by combining two principles in a judicious manner: sparse but more aggressive and dense but more stealthy, forming a stronger but less perceptible attack. Further, by utilizing a network pruning framework, we identify the sensitive locations of the model architecture being trained, which are then targeted by the sparse component of the proposed attack strategy to increase its strength. Finally, through extensive simulations over different NN networks, datasets, and defence mechanisms, we have shown the effectiveness of the proposed hybrid sparse attack approach. To the best of our knowledge, this is the first time in the literature that a Byzantine attack strategy adapts itself opportunistically exploiting the structure of the architecture. We emphasize that the same can be employed in other existing attacks, illustrating yet another weakness of the existing defence mechanisms, and calling for further research in this domain. In the scope of this work, we have limited our focus to extracting architecture-based side information (identifying certain important/sensitive weights) to scale the base adversarial perturbation accordingly and used a proven network pruning algorithm as the base, and we publish our results in [Ozfatura et al., 2024]. As a future research direction, we are planning to design a framework where the direction of the perturbations is also determined with respect to the network architecture while also designing new pruning methods to extract information targeted to the attacking certain parameters.

Finally, we proposed a potential defence mechanism against ROP, called S-CC. By introducing randomness into the clipping process, bucketing the clients, and dynamically choosing a reference point for each bucket, the proposed S-CC mechanism offers complete robustness against ROP, HSA, and HMS while drastically improving the test accuracy in the presence of many other known attacks. However, since it is designed upon the CC aggregator, it also suffers from the curse of dimensionality as such it can achieve sub-optimal test accuracy against attacks that can fully utilize this weakness. In future works, we will address the curse of dimensionality to further increase the robustness of the S-CC.

BIBLIOGRAPHY

[Acar et al., 2021] Acar, D. A. E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. (2021). Federated learning based on dynamic regularization. In *ICLR*.

[Allouah et al., 2023] Allouah, Y., Farhadkhani, S., Guerraoui, R., Gupta, N., Pinot, R., and Stephan, J. (2023). Fixing by mixing: A recipe for optimal byzantine ml under heterogeneity. In *AISTATS*.

[Bagdasaryan et al., 2020] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. (2020). How to backdoor federated learning. In *AISTATS*.

[Baruch et al., 2019] Baruch, G., Baruch, M., and Goldberg, Y. (2019). A little is enough: Circumventing defenses for distributed learning. In *NeurIPS*.

[Bernstein et al., 2019] Bernstein, J., Zhao, J., Azizzadenesheli, K., and Anandkumar, A. (2019). signsgd with majority vote is communication efficient and fault tolerant. In *ICLR*.

[Bhagoji et al., 2019] Bhagoji, A. N., Chakraborty, S., Mittal, P., and Calo, S. (2019). Analyzing federated learning through an adversarial lens. In *ICML*.

[Biggio et al., 2012] Biggio, B., Nelson, B., and Laskov, P. (2012). Poisoning attacks against support vector machines. In *ICML*.

[Blanchard et al., 2017] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In *NIPS*.

[Cao et al., 2021a] Cao, X., Fang, M., Liu, J., and Gong, N. Z. (2021a). Fltrust: Byzantine-robust federated learning via trust bootstrapping. *NDSS*.

[Cao et al., 2021b] Cao, X., Fang, M., Liu, J., and Gong, N. Z. (2021b). Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *NDSS*.

[Carlini and Wagner, 2017] Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee.

[Chen et al., 2020] Chen, X., Chen, T., Sun, H., Wu, S. Z., and Hong, M. (2020). Distributed training with heterogeneous data: Bridging median-and mean-based algorithms. *Advances in Neural Information Processing Systems*, 33:21616–21626.

[de Jorge et al., 2021] de Jorge, P., Sanyal, A., Behl, H., Torr, P., Rogez, G., and Dokania, P. K. (2021). Progressive skeletonization: Trimming more fat from a network at initialization. In *ICLR*.

[El-Mhamdi et al., 2020] El-Mhamdi, E.-M., Guerraoui, R., Guirguis, A., Hoang, L. N., and Rouault, S. (2020). Genuinely distributed byzantine machine learning. PODC '20. Association for Computing Machinery.

[El Mhamdi et al., 2018] El Mhamdi, E. M., Guerraoui, R., and Rouault, S. (2018). The hidden vulnerability of distributed learning in byzantium. In *ICML*.

[El-Mhamdi et al., 2021] El-Mhamdi, E.-M., Guerraoui, R., and Rouault, S. (2021). Distributed momentum for byzantine-resilient stochastic gradient descent. In *ICLR*.

[Evci et al., 2020] Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. (2020). Rigging the lottery: Making all tickets winners. In *ICML*.

[Fang et al., 2020] Fang, M., Cao, X., Jia, J., and Gong, N. Z. (2020). Local model poisoning attacks to byzantine-robust federated learning. In *USENIX Conference on Security Symposium*.

[Farhadkhani et al., 2022] Farhadkhani, S., Guerraoui, R., Gupta, N., Pinot, R., and Stephan, J. (2022). Byzantine machine learning made easy by resilient averaging of momentums. In *ICML*.

[Frankle and Carbin, 2019] Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*.

[Fung et al., 2020] Fung, C., Yoon, C. J., and Beschastnikh, I. (2020). The limitations of federated learning in sybil settings. In *Usenix RAID*.

[Gao et al., 2022] Gao, L., Fu, H., Li, L., Chen, Y., Xu, M., and Xu, C.-Z. (2022). Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *CVPR*.

[Geiping et al., 2020] Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. (2020). Inverting gradients - how easy is it to break privacy in federated learning? In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

[Gorbunov et al., 2023] Gorbunov, E., Horváth, S., Richtárik, P., and Gidel, G. (2023). Variance reduction is an antidote to byzantines: Better rates, weaker assumptions and communication compression as a cherry on the top. In *ICLR*.

[Gupta et al., 2022] Gupta, A., Luo, T., Ngo, M. V., and Das, S. K. (2022). Long-short history of gradients is all you need: Detecting malicious and unreliable clients in federated learning. In *ESORICS*.

[Gupta and Vaidya, 2020] Gupta, N. and Vaidya, N. H. (2020). Fault-tolerance in distributed optimization: The case of redundancy. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, PODC '20, page 365–374, New York, NY, USA. Association for Computing Machinery.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[He et al., 2022] He, L., Karimireddy, S. P., and Jaggi, M. (2022). Byzantine-robust decentralized learning via self-centered clipping. *ArXiv*.

[Jin et al., 2024] Jin, R., Liu, Y., Huang, Y., He, X., Wu, T., and Dai, H. (2024). Sign-based gradient descent with heterogeneous data: Convergence and byzantine resilience. *TNNLS*.

[Karimireddy et al., 2021] Karimireddy, S. P., He, L., and Jaggi, M. (2021). Learning from history for byzantine robust optimization. In *ICML*.

[Karimireddy et al., 2022] Karimireddy, S. P., He, L., and Jaggi, M. (2022). Byzantine-robust learning on heterogeneous datasets via bucketing. In *ICLR*.

[Karimireddy et al., 2020] Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. (2020). Scaffold: Stochastic controlled averaging for federated learning. In *ICML*.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *ICLR*.

[Konečnỳ et al., 2016] Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *NIPS workshop on Private Multiparty Machine Learning*.

[Krizhevsky et al.,] Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research).

[LeCun and Cortes, 2010] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

[Lee et al., 2019] Lee, N., Ajanthan, T., and Torr, P. (2019). SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *ICLR*.

[Li et al., 2019] Li, L., Xu, W., Chen, T., Giannakis, G. B., and Ling, Q. (2019). Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1544–1551.

[Li et al., 2020] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450.

[Liu et al., 2023] Liu, Y., Chen, C., Lyu, L., Wu, F., Wu, S., and Chen, G. (2023). Byzantine-robust learning on heterogeneous data via gradient splitting. In *ICML*.

[McMahan et al., 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *AISTATS*.

[Minka, 2000] Minka, T. (2000). Estimating a dirichlet distribution.

[Muñoz-González et al., 2019] Muñoz-González, L., Co, K. T., and Lupu, E. C. (2019). Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125*.

[Nesterov, 1983] Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Proceedings of the USSR Academy of Sciences*.

[Ozfatura et al., 2021] Ozfatura, E., Ozfatura, K., and Gündüz, D. (2021). Fedadc: Accelerated federated learning with drift control. In *ISIT*, pages 467–472.

[Ozfatura et al., 2024] Ozfatura, E., Ozfatura, K., Kupcu, A., and Gunduz, D. (2024). Aggressive or imperceptible, or both: Network pruning assisted hybrid byzantines in federated learning.

[Peng et al., 2022a] Peng, J., Li, W., and Ling, Q. (2022a). Variance reduction-boosted byzantine robustness in decentralized stochastic optimization. In *ICASSP*.

[Peng et al., 2022b] Peng, J., Wu, Z., Ling, Q., and Chen, T. (2022b). Byzantine-robust variance-reduced federated learning over distributed non-i.i.d. data. *Information Sciences*.

[Pillutla et al., 2022] Pillutla, K., Kakade, S. M., and Harchaoui, Z. (2022). Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*.

[Polyak, 1964] Polyak, B. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*.

[Raynal et al., 2023] Raynal, M., Pasquini, D., and Troncoso, C. (2023). Can decentralized learning be more robust than federated learning? *arXiv*.

[Shejwalkar and Houmansadr, 2021] Shejwalkar, V. and Houmansadr, A. (2021). Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*.

[Shejwalkar et al., 2022] Shejwalkar, V., Houmansadr, A., Kairouz, P., and Ramage, D. (2022). Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *IEEE Symposium on Security and Privacy*.

[Sun et al., 2019] Sun, Z., Kairouz, P., Suresh, A. T., and McMahan, H. B. (2019). Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*.

[Sutskever et al., 2013] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *ICML*.

[Tanaka et al., 2020] Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. (2020). Pruning neural networks without any data by iteratively conserving synaptic flow. In *Neurips*.

[Varma et al., 2021] Varma, K., Zhou, Y., Baracaldo, N., and Anwar, A. (2021). Legato: A layerwise gradient aggregation algorithm for mitigating byzantine attacks in federated learning. In *international conference on cloud computing (CLOUD)*.

[Vogels et al., 2019] Vogels, T., Karimireddy, S. P., and Jaggi, M. (2019). Powersgd: Practical low-rank gradient compression for distributed optimization. *Neurips*, 32.

[Wan and Chen, 2021] Wan, C. P. and Chen, Q. (2021). Robust federated learning with attack-adaptive aggregation. *arXiv*.

[Wang et al., 2020] Wang, J., Tantia, V., Ballas, N., and Rabbat, M. G. (2020). Slowmo: Improving communication-efficient distributed SGD with slow momentum. *ICLR*.

[Wu et al., 2020] Wu, Z., Ling, Q., Chen, T., and Giannakis, G. B. (2020). Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Transactions on Signal Processing*.

[Xiao et al., 2017] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *ArXiv*.

[Xie et al., 2020a] Xie, C., Huang, K., Chen, P.-Y., and Li, B. (2020a). Dba: Distributed backdoor attacks against federated learning. In *ICLR*.

[Xie et al., 2020b] Xie, C., Koyejo, O., and Gupta, I. (2020b). Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in Artificial Intelligence*.

[Xie et al., 2019] Xie, C., Koyejo, S., and Gupta, I. (2019). Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *ICML*.

[Xie et al., 2023] Xie, W., Pethick, T., Ramezani-Kebrya, A., and Cevher, V. (2023). Mixed nash for robust federated learning. *TMLR*.

[Yin and Zeng, 2023] Yin, C. and Zeng, Q. (2023). Defending against data poisoning attack in federated learning with non-iid data. *IEEE Transactions on Computational Social Systems*.

[Yin et al., 2018] Yin, D., Chen, Y., Kannan, R., and Bartlett, P. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*.

[Zhu and Ling, 2022] Zhu, H. and Ling, Q. (2022). Byzantine-robust aggregation with gradient difference compression and stochastic variance reduction for federated learning. In *ICASSP*.

[Özfatura et al., 2024] Özfatura, K., Özfatura, E., Küpçü, A., and Gunduz, D. (2024). Byzantines can also learn from history: Fall of centered clipping in federated learning. *IEEE Transactions on Information Forensics and Security*, 19:2010–2022.