FedXPro: Bayesian Inference for Mitigating Poisoning Attacks in IoT Federated Learning

Pubudu L. Indrasiri, *Member, IEEE*, Dinh C. Nguyen, *Member, IEEE*, Bipasha Kashyap, *Member, IEEE*, Pubudu N. Pathirana, *Senior Member, IEEE*, and Yonina C. Eldar, *Fellow, IEEE*

***Abstract*—Federated learning (FL) has been envisioned to enable many Internet of Things (IoT) devices to perform large-scale machine learning without sharing raw data, resulting in significant privacy improvements. In a wireless IoT system, FL helps clients to secure their confidential information and achieve improved learning performance. However, the conventional FL architecture is vulnerable to Byzantine workers, possessing the potential to send malicious updates that compromise the accuracy of the global model. Previous studies have proposed various secure aggregation rules and attacker detection techniques to address this issue. However, these techniques exhibit limited effectiveness and may lead to a decrease in accuracy. To overcome these limitations, we propose a Byzantine client detection algorithm called FedXPro by combining the predictive coding/biased competition-divisive input modulation (PC/BC-DIM) neural network and geometric median (GM). Predictive coding (PC) is the core of the PC/BC-DIM architecture, which can perform Bayesian inference by fusing priors and likelihoods to determine posterior distributions. The GM is employed to determine the prior knowledge of legitimate clients to execute the PC/BC-DIM algorithm. During training, the framework calculates the probability distribution for a set of valid clients chosen from the GM. In testing, it attempts to reconstruct the same distribution from other clients concerning prior knowledge, and ultimately, the reconstruction power is utilized to filter the malicious clients. Our extensive simulations demonstrate the superiority of our FedXPro approach over other state-of-the-art methods in terms of accuracy, a guaranteed faster convergence rate, and attack detection under different network settings.**

***Index Terms*—Bayesian inference, Byzantine, federated learning (FL), geometric median (GM), Internet of Things (IoT), predictive coding (PC).**

I. Introduction

THE Internet of Things (IoT) refers to devices, such as mobile phones and computers, that are interconnected over the Internet [1]. Due to the increasingly large amount of data generated by IoT devices in recent years, it is envisaged that the IoT will be augmented by big data analytics and eventually artificial intelligence (AI), which will lead to a new world powered by data-driven AI. Traditionally, AI techniques require centralized data from source devices. However, due to rising issues involved in data privacy and communication constraints, it is challenging to transfer data to a third party (e.g., an aggregator) for model training.

Recently, federated learning (FL) has been introduced as an emerging and promising approach by collaborating several devices to perform AI training without disclosing original data, aiming to protect user privacy and conserve network resources, such as bandwidth [2], [3], [4]. It allows massively distributed deep learning model training with dozens or even thousands of participants in a single session [5]. During the training phase of an FL model, the training data set is disseminated among IoT devices, which may belong to different users or organizations [6]. A participant trains the shared model and then uploads local updates to the centralized server (CS). Then it aggregates model updates from the clients while storing their training data on local storage, avoiding any privacy concerns [7]. Since this cycle continues repeatedly, all clients train the shared model independently. Consequently, FL claims to offer significant privacy benefits over standard consolidated data-driven model training methods, with lesser communication bandwidth needs [2], [8].

However, recent research has shown that model updates can still leak private information, necessitating a more organized risk assessment [9]. It is conceivable that security breaches could occur as a result of inherent vulnerabilities in the existing FL architecture [9], [10]. The standard *FedAvg* does not tolerate even a single Byzantine worker's behavior [11]. In the rare circumstance where an attacker knows all benign clients' local updates, model accuracy could plummet from 100% to 0%. It simply needs to set the attacker update to the inverse of the linear combination of other regular updates to counteract the effect of benign clients. Such attackers must be eliminated from an FL system in a timely manner to avoid fraudulent model corruption and irrelevant incentive and profit allocation to adversarial clients.

A. Related Work

There has been a surge of interest in Byzantine-robust FL in recent years. Numerous studies have suggested defense mechanisms to tackle Byzantine clients in an FL system. Most of these mechanisms could be classified into two main

Manuscript received 22 August 2022; revised 30 May 2023; accepted 7 November 2023. Date of publication 20 November 2023; date of current version 26 March 2024. *(Corresponding author: Pubudu L. Indrasiri.)*

Pubudu L. Indrasiri, Bipasha Kashyap, and Pubudu N. Pathirana are with the School of Engineering, Deakin University, Waurn Ponds, VIC 3216, Australia (e-mail: pranpatidewage@deakin.edu.au; b.kashyap@deakin.edu.au; pubudu.pathirana@deakin.edu.au).

Dinh C. Nguyen is with the Department of Electrical and Computer Engineering, The University of Alabama in Huntsville, Huntsville, AL 35805 USA (e-mail: dinh.nguyen@uah.edu).

Yonina C. Eldar is with the Faculty of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 7610001, Israel (e-mail: yonina.eldar@weizmann.ac.il).

Digital Object Identifier 10.1109/JIOT.2023.3334298

TABLE I
IDENTIFIED LIMITATIONS OF PREVIOUS FL ATTACKER DETECTION METHODS. THE LETTER "P" INDICATES THE WORD "PARTIALLY"

Algorithm	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]
Identify Attackers	✓	✓	p	✓	✓	p	p	✓
Data Poisoning		✓	✓	✓	✓	✓	✓	✓
Model Poisoning	✓	✓	✓					✓
Targeted			✓					
Untargeted	✓	✓	✓	✓	✓	✓	✓	✓
No pre trained model	✓	✓	✓	✓	✓	✓	✓	
No validation dataset	✓	✓	✓			✓	✓	✓
No accuracy drop	✓	✓		✓	✓			✓

categories: 1) secure aggregation and 2) Byzantine worker detection.

1) Secure Aggregation: To counter Byzantine attacks in FL, various aggregation rules have been proposed by previous researchers. The conventional and widely used aggregation algorithms, such as Krum [11], GeoMed [11], Trimmed Mean [12], and Medoid [13], primarily focus on developing a robust aggregator capable of estimating the "central point" of the local model updates obtained from clients. However, such aggregation methods are vulnerable to exploitation by attackers who can bias the central point through malicious updates, as observed in studies, such as [14] and [15]. Furthermore, aggregation methods do not distinguish between malicious and nonmalicious updates. Instead, they strive to tolerate adversarial attacks and attenuate their negative consequences via sophisticated model updating processes that are difficult for attackers to exploit. These methods focus on enhancing the robustness and security of the FL system in the presence of malicious clients.

2) Byzantine Worker Detection: Attacker detection refers to the process of identifying and mitigating potential malicious participants within the FL network. The goal is to detect and prevent attackers who may attempt to inject poisoned data or compromise the integrity of the learning process. Techniques, such as anomaly detection, statistical analysis, and behavior monitoring, are commonly employed to identify suspicious activities and protect against adversarial behavior.

While certain prior studies have achieved effective detection of attackers, several limitations persist, as illustrated in Table I. The first limitation of certain techniques from [18] and [21] is the partial identification of attackers, leading to the underperformance of the global model. Additionally, some strategies solely focus on specific threats, such as model poisoning or data poisoning [24]. Certain tactics concentrate only on targeted [25] or untargeted [22], [26] attacks, resulting in an inaccurate global model. The utilization of a pretrained autoencoder model in [23], [27], and [28] is not suitable for distributed systems as it relies on a supervised ML setting and requires manual classification of benign and malicious updates. This approach increases deployment and maintenance costs. Some approaches [20], [24] violate the goal of FL by using a validation data set on the server, which involves sharing original data with third parties. Incomplete attacker identification or failure to address a wide range of attacks leads to significant accuracy drops in certain approaches [18], [21]. Furthermore, the combined defense method proposed in [29], which includes error rate and loss function-based rejection, also requires a validation set. Malicious clients can deceive the server by providing fake sample sizes, known as a "weight attack," as demonstrated in several studies [30], [31]. Therefore, the limitations mentioned above necessitate the development of a robust framework for detecting attackers.

B. Motivation

Motivated by the highlighted limitations of existing approaches, we propose a robust client selection mechanism using the predictive coding/biased competition-divisive input modulation (PC/BC-DIM) architecture [32] and geometric median (GM). Traditionally, the FL process chooses its clients completely or almost randomly, while our approach leverages partial client selection based on the legitimacy of each client. While simpler methods may be effective in certain scenarios, the PC/BC-DIM neural network provides a more sophisticated approach that can handle a wider range of potential attacks and is less susceptible to false positives. Traditional methods that use historical information may not be effective in detecting malicious clients because they may have yet to exhibit any malicious behavior in the past. Therefore, the proposed method provides a more robust and reliable method for detecting malicious clients in FL.

C. Main Contributions of This Article

This article highlights a set of key contributions as follows.

1) *Robust Client Selection:* We propose a novel robust client selection mechanism, called Federated Extra Protection (FedXPro), for distributed synchronous SGD that is resilient to Byzantine failures with provable convergence.
2) *Algorithmic Implementation:* To the best of our knowledge, our research represents the pioneering utilization of Bayesian inference in a predefined PC/BC-DIM neural network, in conjunction with GM, for client selection in FL.
3) *Capability:* The FedXPro is capable of identifying corrupted updates resulting from both targeted and untargeted poisoning attacks by analyzing the parameter vectors of each client, even with a minimal malicious ratio and multiple attack types that occur in a single FL session.
4) *Extensive Simulations:* We effectively demonstrate the robustness of our methodology by countering a diverse array of poisoning attacks while accurately replicating genuine adversarial behavior. Furthermore, we validate the effectiveness of our approach on client nodes utilizing two distinct Deep Learning architectures, namely, multilayer perceptron (MLP) and convolutional neural network (CNN), and three data sets.

Protocol 1: N Denotes the Remote Client Count. The percentage of clients engaged in each round is indicated by $\mathcal{P}$, $\mathcal{P} \in (0, 1]$

1: ***Initialization***: The initial model parameters are randomly generated by the central server.

2: ***Client Selection***: The CS selects a random set of clients, $\lceil N \times \mathcal{P} \rceil$.

3: ***Transmission***: The central server transmits the initial model parameters to selected clients.

4: ***SGD Update and Upload***: Qualified clients utilize their local data to train the models and subsequently send the updates to the central server.

5: ***Aggregation***: The central server calculates the arithmetic mean of the updates received from individual clients.

6: ***Convergence***: Steps 2 through 5 are repeated until the system obtains its optimal global model.

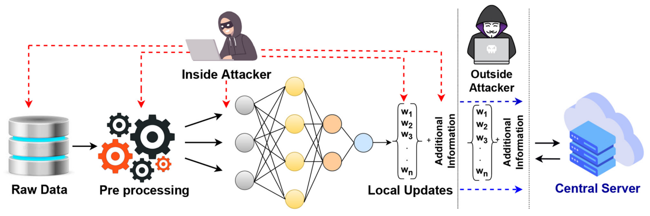

Fig. 1. Overview of the evaluated threat landscape.

D. Overview of This Article

The remainder of this article is organized as follows. Section II explains the problem definition and introduces our defense strategy toward poisoning attacks on FL. Section III illustrates the methods and methodologies used for building the proposed FedXPro. Sections V and VI provide our experimental setting and results with performance evaluations, respectively. Finally, Section VII discusses future directions and conclusions.

II. Federated Learning and Problem Formulation

This section introduces conventional FL and subsequently examines the threat landscape relevant to our proposed solution. We then formulate the problem statement, taking into account the identified threats landscape.

A. Traditional Federated learning

The FL stages are outlined in Protocol 1.

B. Threat Landscape

As shown in Fig. 1, the threat landscape considered in this study involves an attacker who may be a client participating in the FL system or an attacker located between the client and server data transmission. It is assumed that the central server (CS) is honest. The attacker is assumed to have complete control over a portion of the remote clients or the entire FL system, referred to as a white-box attack. That enables the attacker to tamper with various aspects of the system, such as the target training algorithm, source data of legitimate clients, and hyperparameters of the model.

More precisely, this study focuses on Byzantine clients [33], a subgroup of clients attempting to disrupt the effective training of a global model through various Byzantine attacks. These attacks are categorized based on the adversary's goal, with sample reconstruction, information inference, model corruption, and runtime misclassification being key attack goals. This study specifically concentrates on the third and fourth attack goals, primarily achieved through poisoning attacks. Poisoning attacks are divided into two categories: 1) data poisoning and 2) model poisoning attacks. Depending on the attacker's objective, poisoning attacks can be either targeted or untargeted [34]. The data and model poising attacks can be described as follows.

1) Data Poisoning: This involves sophisticatedly tampering with original data points by replacing them with adversarial data points. For example, the label-flipping attack [35] in which the label vector class of a legitimate data set is flipped into a different class.

2) Model Poisoning: This attack type aims to arbitrarily manipulate the gradients from the local SGD before sending them to the CS [14]. The attacker's goal is to corrupt or misclassify the model.

C. Problem Formulation

Our research addresses the question of how to select the most trustworthy clients for step 2 of Protocol 1 without being affected by malicious clients that may try to disrupt the process.

Consider a distributed network under the orchestration of a single CS with a total of N remote clients. Within these clients, a subset B, such that $B \subset N$ is identified as Byzantine clients that exhibit arbitrary behavior. The remaining clients are referred to as honest clients and are represented as $C = N - B$, such that $C \notin B$. Each honest client, denoted as c, possesses a local data set $\mathcal{D}_c$ that comprises S_c data samples. Similarly, a Byzantine client b is associated with a local data set $\mathcal{D}_b$ with S_b data samples. Assume a uniform distribution of data samples among the honest clients. We approach this under the framework of an infinite number of synchronous rounds, represented as T. During round $t \in T$, the CS distributes its parameter vector $\omega^t \in \mathbb{R}^d$ to all clients. Subsequently, every client performs ι iterations of synchronous SGD, a widely used optimization algorithm in AI. In the context of a specific honest client, the loss function for a given sample $\mathcal{S}_c$ is represented as $f(\omega^t, \mathcal{S}_c)$. The ultimate aim is to find the optimal solution for the following average objective function among all honest client devices:

$$F^* = \underset{\omega^t \in \mathbb{R}^d}{\operatorname{argmin}} \; F(\omega) := \frac{1}{N - B} \sum_{c=1}^{N-B} \alpha_c . F_c(\omega) \tag{1}$$

where

$$F_c(\omega) = \frac{1}{S_c} \sum_{\mathcal{S}_c \in D_c} f(\omega^t, \mathcal{S}_c) \tag{2}$$

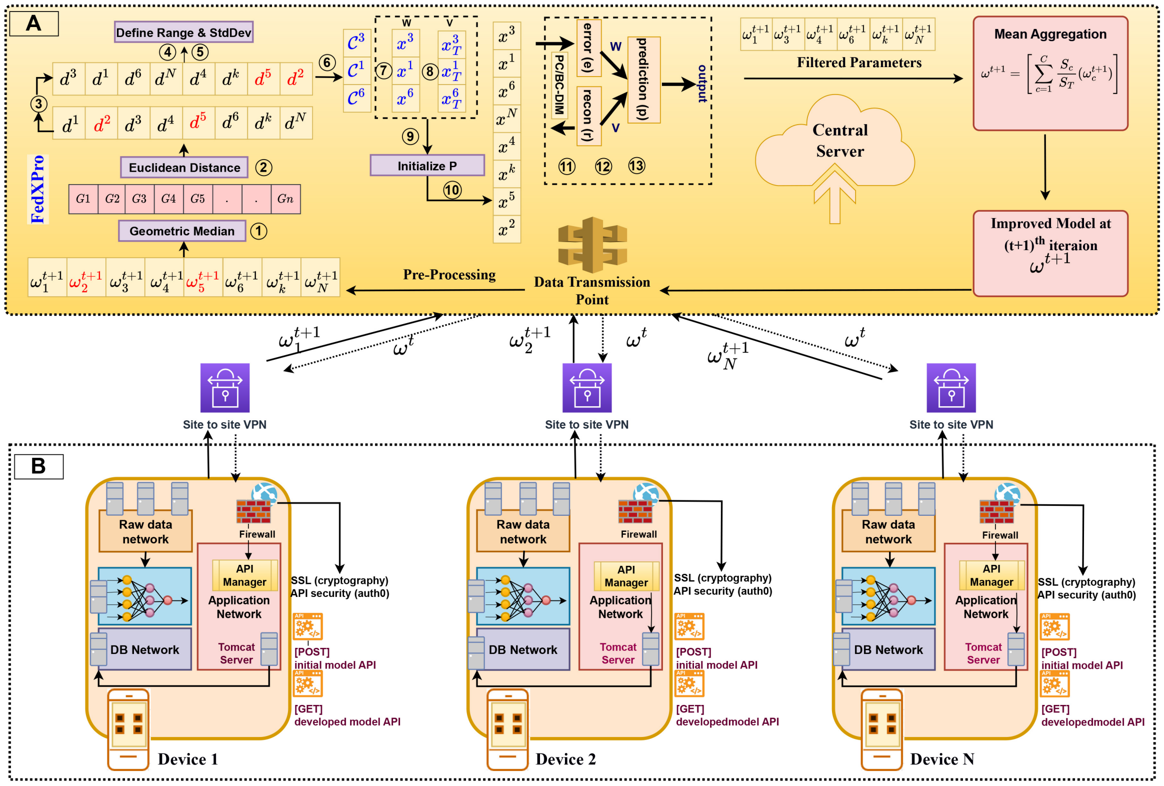

Fig. 2. Illustration of the proposed method, "FedXPro," showcasing the network architecture. Section A represents the CS and its functionality, while Section B depicts the institutions with IoT devices (referred to as "remote clients"). The diagram demonstrates the typical networking flow, starting from the specific IoT devices to the CS, emphasizing the importance of API security.

and $F_c(\omega)$ represents the local objective function for a given honest client and α denotes the relative importance of each client where $\alpha_c \geq 0$ and $\sum_{c=1}^{N-B} \alpha_c = 1$. Since the α_c is user defined, we defined it as $\alpha_c = (S_c/S_T)$ where $S_T = \sum_{c=1}^{N-B} S_c$, the total sample size acquired across all honest devices.

In a potentially vulnerable FL system, each honest client, denoted as c, calculates its gradient update $G_c^t = \nabla f_c(\omega^t, S_c)$. Concurrently, Byzantine clients represented as b, could potentially manipulate data arbitrarily. They propose their own gradient update, represented as G_b^t, reflecting these manipulations. Following these computations, both types of updates are then communicated back to the CS, hence influencing the global model parameters:

$$G_c^t, G_b^t = \begin{cases} \nabla f(\omega^t, S_c), & c \notin B \\ \text{Arbitary}, & i \in B \end{cases} \tag{3}$$

$$\forall c, \quad \omega_c^{t+1} \leftarrow \omega^t - \eta G_c^t \tag{4}$$

$$\forall b, \quad \omega_b^{t+1} \leftarrow \omega^t - \eta G_b^t \tag{5}$$

where η is the predefined learning rate for local ML models. During a given communication round t, the CS aggregates the parameter updates from all clients and updates the current global model using a unique aggregation method

$$\omega_p^{t+1} = \frac{1}{C+B}\left[\sum_{c=1}^{C} S_c(\omega_c^{t+1}) + \sum_{b=1}^{B} S_b(\omega_b^{t+1})\right] \tag{6}$$

where, ω_p^{t+1} denotes the poisoned global model in the server for $t+1$ iteration. Subsequently, the poisoned global model is shared with client devices that lack the ability to perform accurate further training using their own localized data. Hence, the primary challenge lies in identifying and eliminating Byzantine clients B, from the FL system.

III. Proposed Methodology

In this section, we describe our proposed FedXPro system architecture, and assume a *cross-silo* FL scenario in which a limited number of remote clients, like organizations or institutes, engage in the FL process.

A. Overall System Architecture

The proposed FedXPro architecture, depicted in Fig. 2 (A) for the server area and (B) for the remote clients, combines two distinct techniques: 1) GM (in step 1) and 2) PC/BC-DIM neural implementation (in steps 11, 12, and 13). These techniques are integrated to form the proposed algorithm, FedXPro. The following sections will discuss the theoretical foundations of PC/BC-DIM and GM, followed by an explanation of how PC/BC-DIM can be adapted for FL client selection. Subsequently, in Section III-C, a comprehensive workflow will be provided, starting from scratch. Detailed explanations of the employed deep learning architectures (MLP and CNN) on client nodes are provided in Section V-C.

1) PC/BC-DIM Neural Network: The PC/BC-DIM is a type of neural network architecture. It stands for PC/BC-DIM and is based on the predictive coding (PC) theory of brain function [32]. The PC/BC-DIM architecture is designed to

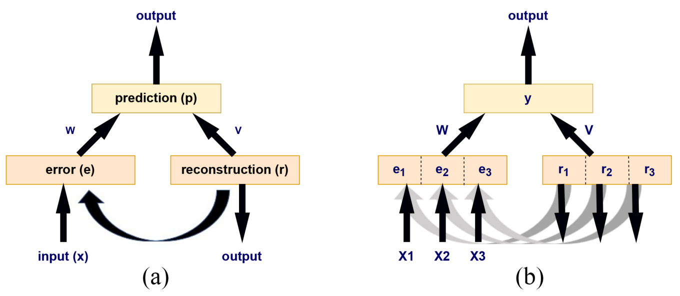

Fig. 3. PC/BC-DIM neural network architecture, (a) for single input, (b) for multiple inputs. Rectangles indicate the three neuron populations (reconstruction, error, and prediction), whereas arrows show connections between them.

learn predictive representations of the input data by iterative processing and updating the input through multiple layers of neurons [32]. This approach is intended to enable the network to encode and store information efficiently while minimizing redundancy.

As in Fig. 3(a), this architecture comprises three separate neural populations, and they constituted a single PC/BC-DIM processing phase while Fig. 3(a) shows when inputs originate from a variety of sources functioning of each neuron population is expressed in the following:

$$\mathbf{r} = \mathbf{V} * \mathbf{p} \tag{7}$$

$$\mathbf{e} = \mathbf{x} \oslash (\mathbf{r} + \epsilon_2) \tag{8}$$

$$\mathbf{p} = (\mathbf{p} + \epsilon_1) \otimes \mathbf{W} * \mathbf{e} \tag{9}$$

where $\mathbf{x}$, $\mathbf{e}$, and $\mathbf{r}$ represent input, error, and reconstruction vectors, respectively, with sizes of $m \times 1$. A specific input's probability distribution is contained in the input vector, and m indicates the number of points in the distribution. The value of m needs to be determined based on the problem statement, which will be explained in Section III-C. The prediction neuron activations are captured by the vector $\mathbf{p}$ of size $n \times 1$, where n indicates the number of inputs. The weight matrix $\mathbf{W}$ is an $n \times m$ matrix, while the feedback weight matrix $\mathbf{V}$ is its normalized transpose, resulting in a size of $m \times n$. The operators $\otimes$ and $\oslash$ represent point-to-point multiplication and division. The value of ϵ_1 is given as $1 * 10^{-6}$ to prevent prediction neurons from becoming nonresponsive, and it also defines the baseline activity rate of prediction neurons. The value of ϵ_2 is equal to $1 * 10^{-4}$, which is used to prevent division by zero errors and determine the required minimum strength of input that affects the response of prediction neurons. Each row of $\mathbf{W}$ serves as a "base vector" or "elementary component" in this context. The weights in the matrix hold the prior knowledge of the external environment. It is important to note that in this case, $\mathbf{W}$ should exclusively consist of probability distributions from the same set, specifically data with binary labels of 1 or 0. Further details on the preparation of values for $\mathbf{x}$, $\mathbf{W}$, and $\mathbf{V}$ will be discussed in Section III-B.

The PC/BC-DIM architecture is listed step-by-step under Algorithm 1. Prediction neuron activations ($\mathbf{p}$) are initialized with small random values or with zero values and (7)–(9) iterate for a number of rounds to update the new $\mathbf{r}$, $\mathbf{e}$ and

Algorithm 1 Activation of *PC/BC − DIM*. **W** Denotes Weights and **x** Denotes Inputs

```
1: PCBCDIM($\mathbf{W}, \mathbf{x}$)
2:     $\mathbf{V} = \hat{\mathbf{W}}^T$
3:     $\mathbf{p} \leftarrow \mathbf{p}$ is initialized as a column vector of zeros
4:     for $i = 1$ : iterations do
5:         $\mathbf{r} \leftarrow \mathbf{V} * \mathbf{p}$
6:         $\mathbf{e} \leftarrow \mathbf{x} \oslash (\mathbf{r} + \epsilon_2)$
7:         $\mathbf{p} \leftarrow (\mathbf{p} + \epsilon_1) \otimes (\mathbf{W} * \mathbf{e})$
8:     end for
9:     return $\mathbf{r}, \mathbf{e}, \mathbf{p} = 0$
```

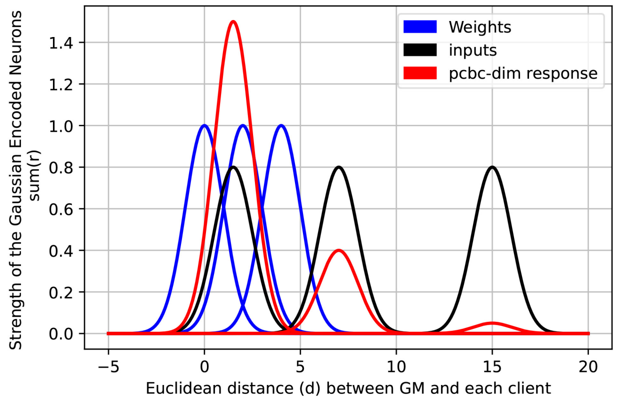

Fig. 4. Probability distributions of priors (blue), inputs (black), and responses (red) of the respective black-colored input.

$\mathbf{p}$ values of the previous $\mathbf{p}$. If the $\mathbf{r}$ becomes constant over time, the network is considered to have converged. The amount of synaptic weights significantly impacts the time it takes to converge. Fig. 4 graphically illustrates the functionality of PC/BC-DIM. It is remarkable that inputs with probability distributions that closely resemble the weights exhibit significantly higher reconstruction power or response, while those with dissimilar distributions exhibit lower power.

B. Adapt PC/BC-DIM in FL Client Selection

To implement the *PC/BC − DIM* architecture, it is necessary to fulfill two distinct conditions as outlined in Algorithm 1.

1) Model updates from individual clients, represented as vectors that contain weights and biases, need to be transformed into a meaningful probability distribution as the input (x).
2) A weight matrix ($\mathbf{W}$) must be prepared from the same set of legitimate or malicious clients.

To achieve this, we begin by converting each parameter vector into a single representative value, enabling the creation of probability distributions for each client. Subsequently, we identify and select a group of reliable clients from which to construct the weight matrix ($\mathbf{W}$). This process mimics the training and building of the model. In this particular case, these two prerequisites were accomplished by using the GM.

1) Geometric Median: A central point that minimizes the total distance to all other points. This is a promising statistical inference strategy for Byzantine resilience on distributed SGD [11], [12], [22]. The *GM* has been employed in previous studies to implement secure aggregation rules in the parameter server since it provides a trustworthy parameter vector across valid and corrupted ones. This idea was utilized to distinguish a subset of genuine clients in order to determine prior knowledge and prepare probability distributions to build **W** for the PC/BC-DIM architecture. The parameter server receives parameter vectors (ω) from clients, and the aim is to find a reliable vector (g^*)

$$g^* = \underset{i \in N}{\text{geomed}}\{\omega_i\} := \underset{g^* \in \mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{N} \|g^* - \omega_i\|. \quad (10)$$

2) Prepare Probability Distributions: Euclidean distances, denoted as $d_i \in \mathbb{R}$, are computed between the optimal parameter vector g^* and each parameter vector ω_i. The distance for a specific client i is represented by d_i

$$d_i\big(g^*, \omega_i\big) = \sqrt{\sum_{a=1}^{q} \big(g_a^* - \omega_{ia}\big)} \quad (11)$$

where q represents the fixed dimension value of the parameter vector that remains consistent across all clients, while a refers to a specific point in both vectors. By calculating the Euclidean distance from g^*, we obtain a meaningful single value for each client. These values can be further transformed into suitable probability distributions, such as a Gaussian distribution, through appropriate conversion techniques.

3) Prepare the Weight (**W**) *Matrix:* With the Euclidean distance and corresponding probability distributions from Section III-B2 available for each client, the next step involves selecting a subset comprising only reliable clients. To accomplish this, we leverage a unique GM property under specific assumptions.

Assumption 1: For a particular FL system, the count of Byzantine workers, designated as B, is less than half the total number of workers, represented by N.

Assumption 2: Each good client's model updates (weights and biases) are close to other good clients.

Assumption 1 is a commonly made assumption in prior Byzantine resilient algorithms, such as [11] and [29]. It serves the purpose of categorizing different client clusters. By considering both Assumptions 1 and 2, we can identify the cluster that encompasses the maximum number of points, indicating the presence of good clients. This information is crucial in determining the prior knowledge, as we now possess the Euclidean distance from g^* to each point.

In Fig. 5 a), the distribution of clients is depicted, with good clients represented by green and bad clients by red color. In Fig. 5 b), the GM point is illustrated. It is apparent that the GM point is situated close to the cluster containing the majority of the points. We can select a subset of good clients by evaluating the distances from the GM to each vector. These good clients are identified based on having the lowest distance values. Consequently, we can utilize the probability distributions of this selected set of good clients to construct the weight matrix **W** (model training), while the remaining vectors are predicted using this trained model.

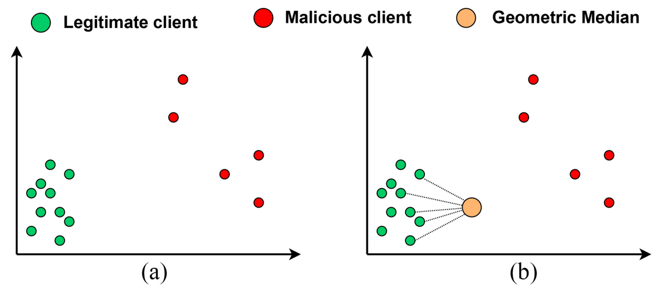

Fig. 5. (a) Distribution of good and bad clients (b) GM is close to the cluster contained maximum clients.

C. FedXPro *Work Flow*

In this section, we provide a step-by-step overview of our proposed algorithm, FedXPro, illustrated in Section III. During the tth global communication round, the CS receives the parameter vectors $(\forall i \in N, \omega_i^t)$ from each client $i \in N$ after their respective local SGD estimations. The remaining procedure can be divided into the following steps.

1) *Step 1:* Using (10), we construct the *GM* vector of all the vectors, ω_t from both the honest (C) and malicious (B) clients.
2) *Step 2:* Applying (11), we compute the Euclidean distances (d) individually between *GM* and each ω_t.
3) *Step 3:* The computed d values are organized in ascending order.
4) *Step 4:* For future calculations of $PC/BC - DIM$, select a range of values (R) between two integers. Ensure that one integer $(R_{\min})$ is smaller than the minimum d, and the other integer $(R_{\max})$ exceeds the maximum d. Next, generate a set of m points uniformly distributed within the selected range R, where m can represent any desired number.
5) *Step 5:* In accordance with Assumption 1, a subset of clients is chosen for further analysis by selecting the initial half or less than half of the clients after arranging the distance values in ascending order in step 2. These selected clients, termed "centers"$(\mathcal{C})$, are considered as prior knowledge for subsequent procedures.
6) *Steps 6 and 7:* To encode each center into the Gaussian distribution, we select the initial half values calculated in step 3 and then proceed to determine the standard deviation σ. Gaussian distribution $(\mathbf{x}_i)$ for center i

$$\mathbf{x}_i = \exp\left(\frac{(R - \mathcal{C}_i)^2}{-2\sigma^2}\right) \quad (12)$$

where

$$\mathbf{W} \leftarrow \forall i \in \mathcal{C}, \mathbf{x}_i. \quad (13)$$

A single dimensional Gaussian encoded input is used to set a weight vector in weight matrix **W**, size of $[\mathcal{C} \times m]$.

After encoding all centers as training weights, the weight matrix $\mathbf{W}$ is

$$\mathbf{W} = \begin{bmatrix} W_1^1 & W_1^2 & W_1^3 & . & . & W_1^m \\ W_2^1 & W_2^2 & W_2^3 & . & . & W_2^m \\ W_3^1 & W_3^2 & W_3^3 & . & . & W_3^m \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ W_C^1 & W_C^2 & W_C^3 & . & . & W_C^m \end{bmatrix}. \quad (14)$$

7) *Step 8:* Next, we define the variable $\mathbf{V}$ as the normalized transpose of $\mathbf{W}$

$$\mathbf{V} = \left[\frac{\mathbf{W}}{\max(\mathbf{W})}\right]^T. \quad (15)$$

8) *Step 9:* Initialize $\mathbf{p}$ with zero values for $PC/BC - DIM$.
9) *Step 10:* The computed d values in step 2 are transformed into a Gaussian distribution using (12). Then PC/BC-DIM is started.
10) *Step 11:* First, calculate the $\mathbf{r}$ values using (7).
11) *Step 12:* Calculate the error $\mathbf{e}$ using (8).
12) *Step 13:* Calculate the prediction $\mathbf{p}$ using (9).
Steps 11, 12, and 13 are iterated up to predefined rounds and finally calculate the reconstruction power by summing the values of vector $\mathbf{r}$. If the summation of reconstruction is very low, that client is identified as a potential attacker, denoted as 0. For the ith client

$$x_i^t \leftarrow \begin{cases} 1, \text{ if } \operatorname{sum}(\mathbf{r}) < 1e^{-6} \\ 0, \text{ otherwise.} \end{cases}$$

Algorithm 2 outlines the complete leveraging of this technique to filter legitimate local updates from legitimate clients. Subsequently, the standard *FedAvg* algorithm is employed to generate an upgraded global model. This process ensures a systematic and reliable approach for incorporating valid local updates into the global model.

The final computational complexity of Algorithm FedEXPro is influenced by three key factors: 1) the computational complexity of PC/BC-DIM; 2) the computational complexity associated with the GM calculation using Weiszfeld's algorithm ($\mathcal{G}$) [36]; and 3) the number of iterations required for convergence. Additionally, the number of clients participating in the FL scenario also plays a role. Thus, the overall computational complexity can be approximated as $O(\mathcal{G} + N \cdot d + \text{iterations} \cdot (N^2 + N + N \cdot E))$, where E determines the size of the weight matrix $\mathbf{W}$ and, N represents the number of clients in the FL scenario.

Considering the involvement of edge devices, CNN architecture with an input layer, two convolutional layers, two hidden layers, and an output layer, the computational complexity can be estimated using symbols, such as $\mathcal{N}$ for the number of input neurons, $\mathcal{H}$ for the height of the input feature map, $\mathcal{W}$ for the width of the input feature map, $\mathcal{F}$ for the number of filters in each convolutional layer, $\mathcal{K}$ for the kernel size of the filters, $\mathcal{D}$ for the number of neurons in the hidden layers, and $\mathcal{M}$ for the number of output neurons. The final computational complexity of the CNN in FL can be approximated as $O(2 \cdot \mathcal{F} \cdot \mathcal{K}^2 \cdot \mathcal{H} \cdot \mathcal{W} + 2 \cdot \mathcal{D} + \mathcal{D} \cdot \mathcal{M})$.

Algorithm 2 FedXPro: Federated Extra Protection

```
Require: $N$: total number of clients
 1: Server-side execution
 2: Initialize parameters: $\epsilon_1$, $\epsilon_2$
 3: Initialize empty sets: $U$, $H$, $M$
 4: for $t \leftarrow 1$ to $T$ do
 5:     for $i \leftarrow 1$ to $N$ do
 6:         $\omega_i^{t+1} \leftarrow \omega^t - \eta\nabla f_i(\omega^t, J_i^t)\}$
 7:         $U \leftarrow U \cup \{\omega_i^{t+1}\}$
 8:         $M \leftarrow M \cup \{\omega_i^{t+1}\}$
 9:     $M \leftarrow$ CalculateGeometricMedian($M$) $\leftarrow$ eq. (10)
10:     $D \leftarrow$ CalculateEuclideanDistances($M, U$) $\leftarrow$ eq. (11)
11:    $D \leftarrow$ SortAscending($D$)
12:     $R_{\max} \leftarrow$ FindMaxThreshold($D$)
13:     $R_{\min} \leftarrow$ FindMinThreshold($D$)
14:     $\mathcal{N} \leftarrow$ generatePoints($R_{\max}, R_{\min}$)
15:     $\sigma \leftarrow$ FindSigma($D$)
16:     $\mathbf{W} \leftarrow$ InitializeWMatrix($\frac{|D|}{2}, \mathcal{N}$)
17:    for $i \leftarrow 1$ to $|U|$ do
18:         $d_i \leftarrow D[i]$
19:         $\mathbf{x} \leftarrow$ CalculateGaussianDistribution($d_i$)
20:         if $i \leq \frac{|U|}{2}$ then
21:             $\mathbf{W} \leftarrow$ UpdateWMatrix($\mathbf{W}, \mathbf{x}$)
22:         end if
23:        $\mathbf{y}, \mathbf{e}, \mathbf{r} \leftarrow$ PCBCDIM($\mathbf{W}, \mathbf{x}$)
24:         if sum($\mathbf{r}$) $\leq$ Threshold then
25:             EliminateClient($i$)
26:         else
27:             $H \leftarrow H \cup \{i\}$
28:         end if
29:     $L \leftarrow$ GetLocalModelUpdates($H$)
30:     $\omega^{t+1} \leftarrow$ FederatedAveraging($L$)
31:     CS broadcasts $\omega^{t+1}$ across clients
32:     Client − side execution
33:    forH $\in N$ in parallel do
34:         $\omega_i^{t+2} \leftarrow \omega^{t+1} - \eta\nabla f_i(\omega_{t+1}, S_i^{t+1})$
35:    end for
36:    Return $\omega^{t+2}$ to server
37: end for
    =0
```

IV. Convergence Analysis

In this section, we demonstrate the convergence properties of FedXPro and its resilience against Byzantine attacks. We leverage prior work by [37] to highlight the effectiveness of biased client selection for faster convergence. Expanding on their findings, we prove the convergence of FedXPro by promptly eliminating clients with higher $F_i(\omega)$ values. This selective favoring of specific clients enhances the method's overall robustness.

A. Assumptions

The following assumptions are made regarding the functions $F_1, F_2, \ldots\ldots, F_n$. The "Softmax classifier" and "logistic regression" are prominent examples of Assumptions 3 and 4. For client $i \in$ [N].

Assumption 3: $F_1, F_2, \ldots, F_i$ are L-smooth: for all $\mathbf{x}$ and $\mathbf{y}$, $F(\mathbf{x}) \leq F(\mathbf{y}) + (\mathbf{x}-\mathbf{y})^T \nabla F_i(\mathbf{y}) + (L/2)\|\mathbf{x}-\mathbf{y}\|^2$.

Assumption 4: $F_1, F_2, \ldots, F_i$ are μ-strongly convex: for all $\mathbf{x}$ and $\mathbf{y}$, $F(\mathbf{x}) \geq F(\mathbf{y}) + (\mathbf{x}-\mathbf{y})^T \nabla F(\mathbf{y}) + (\mu/2)\|\mathbf{x}-\mathbf{y}\|^2$.

Assumption 5: Let ξ_i be the mini-batch, uniformly sampled from each client device i at random. Consequently, an unbiased stochastic gradient is produced, that is $\mathbb{E}[g_i(\mathbf{y}_i, \xi_i)] = \nabla F_i(\mathbf{y}_i)$. Further, the each honest device's stochastic gradients variance is upper bounded. $\|\mathbb{E}[g_i(\mathbf{y}_i, \xi_i)] - \nabla F_i(\mathbf{y}_i)\| \leq \sigma^2$.

Assumption 6: Stochastic gradients have a uniformly bounded expected squared norm, i.e., $\mathbb{E}\|g_i(\mathbf{y}_i, \xi_i)\|^2 \leq G^2$.

Next, present two metrics that will be relevant for convergence analysis.

Definition 1 (Difference of Local, Global Objective Functions): For the global optimum $= \mathbf{y}^* = \text{argmin}_{\mathbf{y}} F(\mathbf{y})$ and for local optimum $= \mathbf{y_i}^* = \text{argmin}_{\mathbf{y}} F_i(\mathbf{y})$. The difference ($\hbar$) is defined as

$$\hbar = \sum_{i=1}^{N} \alpha_i \big(F_i(\mathbf{y}^*) - F_i(\mathbf{y}_i^*)\big) \geq 0. \tag{16}$$

Here, the $\hbar$ denotes a crucial characteristic of the local and global optimum optimization functions. It indicates a more diverse data set if the value is high. If it is at zero, then both the local and global levels are steady.

In contrast to random client selection with unbiased manner, the FedXPro leverages biased client selection. The FedXPro selects clients using the reconstruction power ($\mathbf{r}$) which leads to select clients having higher $F_i(\mathbf{y})$. This approach impacts the global and local objective differences due to higher $\sum_{i=1}^{N} \alpha_i (F_i(\mathbf{y}^*))$. The next definition illustrates the effect of the FedXPro for the objective difference.

Definition 2 (Selection Bias): Let us take the set of selected clients B using FedXPro with respect to current global model $\mathbf{y}$ as $B(\Theta, \mathbf{y})$, where Θ denotes the FedXPro. For any attacker $b \in B$, we define a function $f(B(\Theta, \mathbf{y}), \mathbf{y}')$

$$\frac{\left[\frac{1}{[B]} \sum_{b \in B} \big(F_b(\mathbf{y}') - F_b^*\big)\right]}{F(\mathbf{y}) - \sum_{i=1}^{N}} \geq 0. \tag{17}$$

This function reflects the bias of the approach FedXPro. Here, $\mathbf{y}'$ denotes the local models that are evaluated. As in study [37], we now define two new distinct measurements $(\bar{f}, \tilde{f})$, where $(\bar{f} \leq \tilde{f})$ shown in (18), at the bottom of the page, which are independent from $\mathbf{y}$ and $\mathbf{y}'$. These measurements allow us to set an error limit for the convergence study.

In FedXPro, we use a biased client selection approach which is to eliminate the client updates which have a higher loss. Hence, the $\bar{f}$ and $\tilde{f}$ get higher values. In our analysis, we prove larger $\bar{f}$ signifies faster convergence, despite the possibility of an error gap.

B. Convergence Result

The convergence results for FedXPro for standard *FedAvg* with limited device participation are shown here regarding the local-global optimum objective difference.

Theorem 1: Under the conditions of Assumptions 3 to 6, let $\gamma = (8L/\mu)$ and the learning rate $\eta_t = [2/\mu(\gamma + t)]$. After T iterations of FedXPro with only selected devices participating, the error is satisfied by (19), shown at the botttom of the page.

Proof: See Appendix A. ■

One important conclusion from Theorem 1 is that a higher $\bar{f}$ causes a faster convergence rate, as $\mathcal{O}$ $(1/[T + \bar{f}])$.

V. Experiments

In the experimental setting of our proposed FedXPro algorithm, we employ diverse real-world data sets and utilize two different neural networks. Additionally, we conduct a series of poisoning attacks that replicate genuine adversarial behavior. Our research utilized a Windows 10 system with an Intel i7-6500U CPU (two cores and four logical processors), 8-GB DDR3 RAM, and third-party tools Jupyter Notebook and PyCharm.

A. Data Sets

For the simulations, we employ three distinct data sets for an image classification task.

1) MNIST: This well-known data set of handwritten digits has 10 000 samples in the test set and 60 000 examples in the training set. Each illustration is 28×28 pixels in size and includes a label. All values are normalized to [0, 1], and the range of the pixel values is 0 to 255.

2) CIFAR-10: This comprises 60 000 32×32 color photographs divided into ten classes, each with 6000 samples. In addition, all pixel values are normalized to [0, 1].

3) FMNIST: This contains 70 000 fashion products in 28×28 grayscale photographs from ten categories, with 7000 images per category.

B. Data Distribution Over Clients

The data set was partitioned among a total of 50 clients, comprising 30 legitimate clients denoted as C and 20 Byzantine clients denoted as B. This partitioning aimed to account for adversarial behavior within the system. To introduce data heterogeneity, an uneven distribution strategy was employed, assigning a varying number of samples from the

$$\bar{f} = \min_{\mathbf{y},\mathbf{y}'} f\big(B\big(\Theta, \mathbf{y}'\big)\big), \quad \tilde{f} = \min_{\mathbf{y},\mathbf{y}'} f\big(B\big(\Theta, \mathbf{y}^*\big)\big). \tag{18}$$

$$\mathbb{E}[F(\bar{\mathbf{y}})^{(T)}] - F^* \leq \underbrace{\frac{1}{(T+\gamma)}\Big[\frac{4L(32\pi^2 G^2 + \sigma^2/B}{3\mu^2 \bar{f}} + \frac{8L^2\hbar}{\mu^2} + \frac{L\gamma\|\overline{w}^{(0)} - w^*\|^2}{2}\Big]}_{E1} + \underbrace{\frac{8L\hbar}{3}\Big(\frac{\tilde{f}}{\bar{\bar{f}}} - 1\Big)}_{E2}. \tag{19}$$

training set to each client. Specifically, the number of samples allocated to individual clients ranged between 100 and 1500.

C. Neural Network Training

For the MNIST data set, we utilize a MLP with two hidden layers, employing ReLU activation in the network and Softmax activation in the output layer. On the other hand, for the FMNIST data set, we utilize a CNN comprising two 5×5 convolution layers with 20 and 50 channels, respectively, followed by max pooling. It also includes a fully connected layer with 500 units, ReLU activation, and a Softmax output layer. For the more challenging CIFAR-10 data set, we employ a heavier CNN model with two 5×5 convolution layers having 64 channels each. This is followed by two fully connected layers with 384 and 192 units, respectively, 2×2 max pooling, and a Softmax output layer.

D. Poisoning Attacks and Malicious Ratio (ρ)

We address data and model poisoning attacks that are both targeted and untargeted. The following attack types were considered, and Byzantine data sets were generated accordingly. The first two attacks are data poisoning, whereas the latter is model poisoning. The noise attacks were constructed with the *Python skimage* package, and model attacks were created with the *PyTorch torchattacks* library.

1) *Label Flipping (LF) Attack [35]:* The attacker swaps the labels of all original data samples from one class to another, like when the label "patient" is replaced with the label "control" in a patient and control classification.
2) *Noise Injecting Attacks [38]:* Noise insertion is a masking technique for modifying an original format of data into a different one. Each noise source was put to the test by increasing and decreasing its amplitude. We experimented with four noise attacks. Let ϕ and ϕ^p denote the clean and perturbed image, respectively, and τ denote the noise. Then, $\Phi^p = \Phi + \tau\Phi$ where τ depends on (μ, σ), that are used to control the noise characteristics, where μ represents the mean, and σ represents the standard deviation of the noise distribution. For *Gaussian noise*, attacker $b \in B$ generates its model ω_b^t with $\mu = 0$ and different σ values. For *random noise*, the noise was added randomly by altering the μ and σ. For *salt and pepper noise*, is a random disturbance that corrupts images with scattered white and black pixels, degrading image quality. For *Speckle noise*, this arises during image capture due to external factors affecting the sensor, causing changes in both μ and σ.
3) *Fast Gradient Sign Attack (FGSM) [39]:* This is a gradient-based attack. Perturbed images were generated using the gradient $\nabla_\Phi f_i(\omega^t, S)$. $\Phi^p = \Phi + \epsilon sign(\nabla_\Phi f_i(\omega^t, S))$ where ϵ denotes perturbed amount.
4) *Projected Gradient Descent (PGD) [39]:* This is a white box attack that allows an attacker to access the model gradients. $\Phi^p = \Pi_{\Phi+S}(\Phi + \alpha \text{sign}(\nabla_\Phi f_i(\omega^t, S)))$.

TABLE II
PERFORMANCE COMPARISON OF ALGORITHMS ON MNIST DATA SET UNDER LF ATTACK IN TWO SCENARIOS

Algorithm	Attack Strategies	
	Single Label	Multi Label
LoMar	91.2	88.5
Krum	88.3	73.4
Median	65.5	65.5
FG	78.3	80.2
FG + Krum	81.2	41.5
FedAvg(No Attack)	93.1	93.1

VI. RESULTS

This section summarizes the results and theoretical findings of the proposed algorithm, FedXPro.

Before diving into our findings, we noticed that the study [40] compared their approach, LoMar, with other state-of-the-art techniques in the context of label-flipping attacks. The comparison was conducted in two scenarios: 1) single-LF, where only one label is flipped at a time and 2) multi-LF, where multiple labels are flipped simultaneously. To establish a fair benchmark, we adopt the techniques used in that comparison, as listed in Table II, to evaluate the performance of our proposed algorithm, FedXPro, under the same attack setting. According to the information given in Table II, the highest accuracy score was obtained with no attack condition in both scenarios, which implies when the standard *FedAvg* execution in a Byzantine-free environment.

Their analysis, as presented in Table II, reveals that each algorithm experiences a decline in accuracy when generating the global model securely. By excluding the "No Attack" condition, it is evident that LoMar achieves the highest accuracy under the LF attack in both single-label and multilabel scenarios. However, it demonstrates a significant drop in accuracy in the multilabel scenario. This comparison provides a clear context for conveying our research findings and highlighting the robustness of FedXPro. Our approach focuses on eliminating all malicious client updates, creating a Byzantine-free environment, and subsequently executing the standard *FedAvg* algorithm. In Section VI-A, we demonstrate that our approach achieves accuracy levels comparable to *FedAvg* in the absence of attacks, even with the presence of malicious clients, in both label-flipping scenarios. Furthermore, our method outperforms other attacks outlined in Section IV.

A. Robustness of FedXPro

This section presents a detailed analysis of the robustness of FedXPro. Using three data sets, we compare the classification accuracy of FedXPro under a Byzantine setting with two related schemes. In addition, we compared our method to GeoMed [11], which we used as a prerequisite. This allowed us to assess our method's performance and effectiveness in relation to GeoMed. The first scheme involves the performance of traditional *FedAvg* under a Byzantine setting (*FedAvg* With Attack). In contrast, the second scheme focuses on *FedAvg* without a Byzantine setting *FedAvg (No Attack)*. The Byzantine setting encompasses all the attacks mentioned in Section IV, individually or in combination. For FedXPro, we denote it as FedXPro *(With Attacks)*,

TABLE III
PERFORMANCE CONCERNING DIFFERENT ALGORITHMS AND DATA SETS UNDER CONSIDERED MAXIMUM ρ FOR EACH ATTACK

Algorithm	Datasets		
	MNIST	FMNIST	CIFAR-10
FedXPro *(With Attacks)*	94.12 $\pm$0.3	91.45 $\pm$0.3	64.68 $\pm$0.3
GeoMed (With Attacks)	89.50 $\pm$0.3	83.15 $\pm$0.5	51.32 $\pm$0.4
FedAvg (Label Flipping)	79.21	81.12	22.67
FedAvg (Random)	79.21	12.34	10.04
FedAvg (Gaussian)	83.28	77.37	40.56
FedAvg (Speckle)	86.65	75.76	48.32
FedAvg (Salt&Pepper)	82.71	85.12	48.21
FedAvg (FGSM)	72.43	81.55	52.92
FedAvg (PGD)	84.21	71.65	44.36
FedAvg (No-Attack)	94.97	91.98	64.97

specifically FedXPro *(LF)*, FedXPro *(Random)*, FedXPro *(Gaussian)*, FedXPro *(Sperckle)*, FedXPro *(Salt&Pepper)*, or FedXPro *(LF, Random, Gaussian, etc.)*.

1) Comparison of Algorithms for Each Data Set: In this analysis, we conducted experiments with a single attack type during each FL training session without mixing multiple attacks. As shown in Table III, it is evident that FedXPro *(With Attacks)* achieves comparable accuracy to *FedAvg (No Attacks)* across all three data sets, for both data and model poisoning attacks.

The accuracy score for FedXPro *(With Attack)* is reported as the mean value with a margin of $\pm$ 0.3, representing the range of accuracy values across all attacks. Our analysis amply demonstrates that *FedAvg (With Attack)* experiences a significant drop in accuracy for all attacks on the MNIST, FMNIST, and CIFAR-10 data sets. In contrast, FedXPro *(With Attacks)* achieves accuracy values of 94.12% $\pm$ 3%, 91.45% $\pm$ 3%, and 65.68% $\pm$ 3% for the respective data sets. Specifically, *FedAvg (LF)* yields 79.21%, 81.12%, and 22.67% accuracy, exhibiting a similar pattern for other attacks. Consequently, this analysis reveals a substantial improvement in performance compared to the benchmark schemes in Table III. Notably, *FedAvg (No Attack)* achieves accuracy values of 94.97%, 91.98%, and 64.97%, further highlighting the superiority of FedXPro in comparison.

2) Multiple Attack Types in a Single FL Training Process: In this analysis, we consider the scenario where one or more attacks are combined to initiate an attack in each round or specific rounds. We conducted experiments with three different attack combinations: 1) LF, Random, Gaussian noise, and Speckle, representing a mixture of data poisoning attacks; 2) FGSM and PGD, representing a mixture of model poisoning attacks; and 3) LF, Random, Gaussian, and FGSM, representing a combination of both data and model poisoning attacks. The results, presented in Table IV, reveal a significant accuracy drop in FMNIST and CIFAR-10 data sets under this setting, while the MNIST data set exhibits a moderate decline in accuracy.

We further computed additional performance metrics to the MNIST data set. The results are presented in Table V.

Based on the results of this analysis, it is evident that our proposed approach, FedXPro, consistently outperforms other methods. FedXPro achieves nearly identical accuracy to

TABLE IV
PERFORMANCE WITH RESPECT TO DIFFERENT ALGORITHMS AND DATA SETS UNDER CONSIDERED MAXIMUM ρ FOR EACH ATTACK

Algorithm	Datasets		
	MNIST	FMNIST	CIFAR-10
FedXPro *(With Attacks)*	94.12 $\pm$0.3	91.45 $\pm$0.3	64.68 $\pm$0.3
GeoMed (With Attacks)	90.35 $\pm$0.3	86.50 $\pm$0.4	49.50 $\pm$0.4
FedAvg (withattacks-1)	86.67	74.61	42.65
FedAvg (withattacks-2)	78.87	73.83	41.98
FedAvg (withattacks-3)	85.33	69.21	43.22
FedAvg (No-Attack)	94.37	91.98	64.97

TABLE V
PERFORMANCE METRICS (PRECISION, RECALL, AND F1-SCORE) FOR DIFFERENT ALGORITHMS ON THE MNIST DATA SET UNDER THE CONSIDERED MAXIMUM ρ FOR EACH ATTACK

Algorithm	MNIST		
	Precision	Recall	F1-Score
FedXPro *(With Attacks)*	93 $\pm$0.3	94 $\pm$0.3	93 $\pm$0.5
FedAvg (withattacks-1)	85.32	84.61	84.54
FedAvg (withattacks-2)	72.81	74.43	79.21
FedAvg (withattacks-3)	81.44	81.21	83.43
FedAvg (No-Attack)	93.82	94.91	93.91

FedAvg (No Attacks) for all three data sets, even when subjected to multiple attacks. Moreover, the accuracy of FedXPro remains consistent with that of *FedAvg (No Attacks)* across various communication rounds, highlighting the robustness and stability of our approach.

3) Comparison With Different Malicious Ratio (ρ): We introduce a parameter, denoted as ρ, to control the level of corruption for each attack. We set ρ values for noise attacks as [0.05, 0.25, 1, 5, 10, 20, 30, 50, 60], representing different σ^2 values. For FGSM and PGD attacks, we use ρ values as [0.001, 0.01, 0.1, 0.5, 0.9], which correspond to distinct ϵ and α values, as defined in Section IV. The ρ values for LF attacks are configured from 20% to 100%, where 20% represents a single-label LF attack, and higher values indicate multilabel attacks where all labels of a particular data set are flipped into other labels.

Fig. 6 illustrates the performance of FedXPro when increasing the malicious ratio of participating attackers for a single attack type under data poisoning attacks. Similarly, Fig. 7 demonstrates the performance under model poisoning attacks. Notably, FedXPro *(With Attacks)* consistently achieves the highest test accuracy for all three data sets. Moreover, it maintains this accuracy even when increasing the ρ values for both data and model poisoning attacks. In contrast, *FedAvg (With Attacks)* is surpassed by the attacks as the ρ value increases. Additionally, the performance of FedXPro is affected by increasing σ^2 values in noise attacks, while it becomes more robust when decreasing ϵ and α values in model attacks.

B. Convergence Simulation

We visually depict FedXPro convergence and the effect of eliminating Byzantine workers on resilience. Our experiments comprised four categories with 200 communication rounds (t).

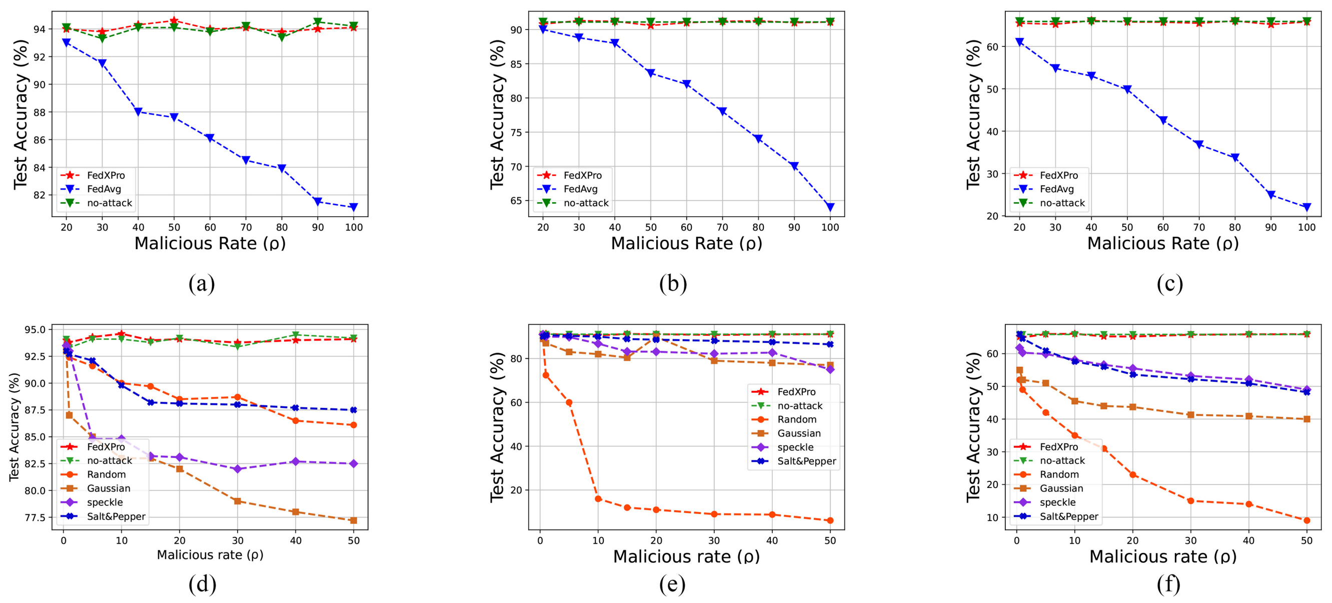

Fig. 6. Overall Test Accuracy for LF Attack and Noise Injection Attacks (Gaussian, Random, Salt&Pepper, and Spearkle) on Three Data sets (MNIST, FMNIST, and CIFAR-10). The First Row Shows Flipping Attack Results and the Second Row Shows Noise Attacks. For Flipping Attack, Malicious Ratio $\rho = [20, 40, 60, 80, 100]$ and for Each Noise Attack $\rho = [0.5, 1, 5, 10, 15, ..,40, 50]$. (a) and (d) MNIST. (b) and (e) FMNIST. (c) and (f) CIFAR-10.

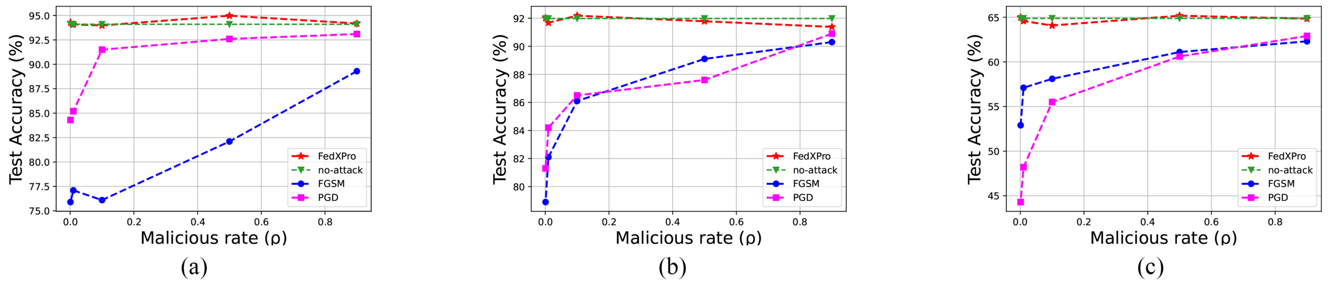

Fig. 7. Overall test accuracy for FGSM and PGD attacks on three data sets (MNIST, FMNIST, and CIFAR-10). The malicious ratio $\rho = [0.001, 0.01, 0.1, 0.5, 0.9]$ for each attack. (a) MNIST. (b) FMNIST. (c) CIFAR-10.

Fig. 8 (a)–(c) presents the convergence analysis under LF attacks, while subfigure (d) represents convergence under multiple attacks. We compared the convergence of FedXPro *(No Attacks)* with *FedAvg (With Attacks)*, and *FedAvg (No Attacks)*. Notably, FedXPro *(With Attacks)* exhibited a similar convergence rate to *FedAvg(No Attacks)*, whereas *FedAvg (With Attacks)* displayed inconsistent behavior. FMNIST showed more variation with attacks, while MNIST exhibited smooth convergence, albeit reaching the desired accuracy level earlier.

C. Attacker Detection Rate Analysis

Throughout this study, we assume that the attacker count is less than half of the total clients. Previous analysis (Section VI-A1) demonstrated that under this assumption, our proposed algorithm outperforms in scenarios involving single and multiple attacks. However, an interesting observation arises in the case of multiple attacks, as depicted in Fig. 9(a). Here, the attacker employs various attack types, and the total number of attacked clients equals or surpasses the number of nonattacked clients. Notably, a substantial cluster predominantly consists of nonattacked clients, enabling FedXPro to detect attackers to some extent. However, with an increasing number of attackers, the detection rate experiences a sharp decline.

In contrast, when an attacker launches a single attack type, and the number of attacked devices exceeds the number of nonattacked clients, the detection rates drop to zero, as shown in Fig. 9(b). The lower row plot in Fig. 9 illustrates the decreasing detection rate as the number of attackers increases in single and multiattack scenarios. Notably, after surpassing the threshold point of half the number of clients, the attacker detection rate undergoes a significant and rapid decline.

VII. Conclusion

In this study, we have explored a novel mechanism called FedXPro for detecting Byzantine attackers in the context of FL. By combining the Bayesian inference with the PC/BC-DIM neural network and *GM*, we have addressed several limitations of traditional defense methods. The experimental results demonstrate the superior performance of our proposed approach compared to conventional methods.

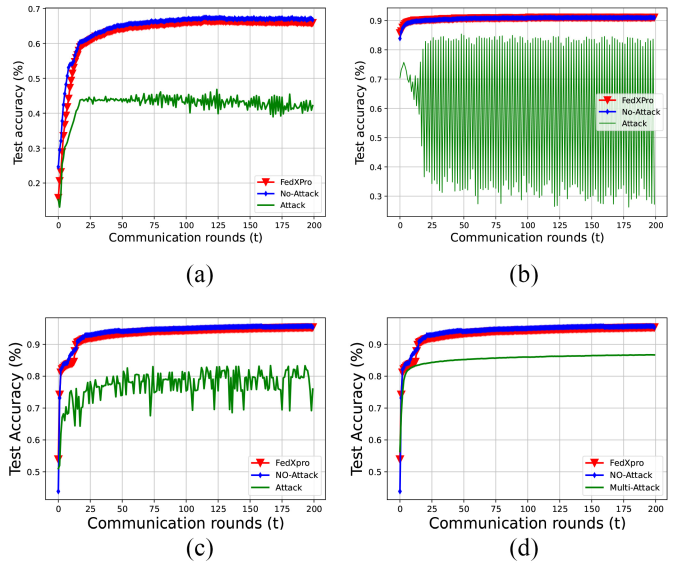

Fig. 8. Convergence analysis of FedXPro with respect to three data sets under LF attack and convergence of MNIST deadset under multiple attacks. (a) CIFAR-10 (single). (b) FMNIST (single). (c) MNIST (single). (d) MNIST (multi).

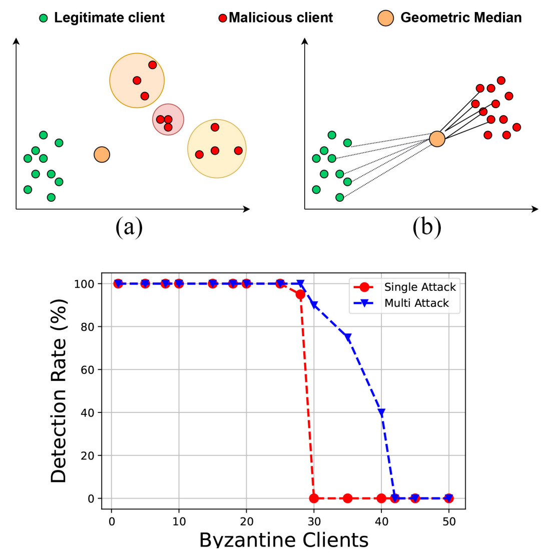

Fig. 9. Upper row, a) GM point with multiple attacks indicates by each colored circle, b) GM point biased to malicious clients. The lower row plot indicates the attacker detection rate.

Moving forward, our future work will focus on optimizing the speed of FedXPro, as it operates similarly to a neural network and requires predefined iterations for each client. Regarding client selection, we aim to develop a robust selection framework that goes beyond detecting only malicious updates. Additionally, we will consider additional factors, such as heterogeneous devices and diverse memories on client devices, while enhancing the PC/BC-DIM as a versatile utility for multicriterion client selection.

Appendix A
Proof of Geometric Median

Lemma 1: Let $x_1, x_2, \ldots, ., x_d$ be a set of d points with each $x_i \in \mathbb{R}^n$, the GM is defined as

$$\arg\min_{y\in\mathbb{R}} \sum_{i=1}^{d} \|x_i - y\|_2 \tag{20}$$

where argmin is the point y from where the sum of all distances of $x_i^{\prime s}$ is minimum.

Proof: Let

$$f(y) = \sum_{i=1}^{d} |x_i - y|.$$

If we rearrange $x_{i's}$ as $x_1 < x_2 < \ldots < x_d$ then

$$\sum_{i=1}^{d} |x_i - y| = \sum_{i=1}^{d-1} |x_i - y| + (x_d - x_1)$$

when $y \in [x_i, x_n]$, then

$$\begin{aligned} f(y) &= \sum_{i=1}^{d} |x_i - y| = |x_{\frac{d+1}{2}} - y| \\ &\quad + (x_d - x_1) + (x_{d-1} - x_1) + \cdots \\ &\quad + \left(x_{\frac{d+3}{2}} - x_{\frac{d-1}{2}}\right) \\ f(y) &= |x_{\frac{d+1}{2}} - x| + \text{constant}. \end{aligned} \tag{21}$$

This is the absolute value function with vertex $(x_{[d+1/2]}, \text{Constant})x_{(d+1/2)}$minimize $f(y)$

$$\text{If } f(y) = \sum_{i=1}^{d} |x_i - y| = |x_{\frac{d}{2}} - y| + |x_{\frac{d+1}{2}} - y| + \text{constant}. \tag{22}$$

The minimum occurs at $[df(y)/dy] = 0$

by differentiating and setting $f'(y)$ to zero, we get

$$\frac{|x_{\frac{d}{2}} - y|}{\left(x_{\frac{d}{2}} - y\right)} + \frac{|x_{\frac{d+2}{2}} - y|}{\left(x_{\frac{d+2}{2}} - y\right)} = 0. \tag{23}$$

$$x = \frac{x_{\frac{d+2}{2}} + x_{\frac{n}{2}}}{2}$$

$$\sim \text{The median is halfway between } x_{\frac{d+2}{2}} \text{ and } x_{\frac{d}{2}}. \tag{24}$$

$$x_{\frac{d}{2}} - x = \left(x_{\frac{d+2}{2}} - x\right). \tag{25}$$

Therefore

$$\frac{|x_{\frac{d}{2}} - x|}{\left(x_{\frac{d}{2}} - x\right)} + \frac{|x_{\frac{d}{2}} - x|}{-\left(x_{\frac{d}{2}} - x\right)} = 0. \tag{26}$$

Then x point is minimum to the GM. ■

Appendix B
Ground Works for Proof of Theorem 1

We illustrate the lemmas used for proof of Theorem 1. Here, B denotes the selected clients.

Lemma 2: If F_i is $L-$ smooth and has a global minimum at $\mathbf{y}_i^*$, then we have that for any $\mathbf{y}_i$ in F_i's domain

$$\|\nabla F_i(\mathbf{y}_i)\|^2 \le 2L\left(F_i(\mathbf{y}_i) - F_i\left(\mathbf{y}_i^*\right)\right). \tag{27}$$

Proof:

$$F_i(\mathbf{y}_i) - F_i(\mathbf{y}_i^*) - \langle \nabla F_i(\mathbf{y}_i^*), \mathbf{y}_i - \mathbf{y}_i^* \rangle \geq \frac{1}{2L}\left\|\nabla F_i(\mathbf{y}_i) - \nabla F_i(\mathbf{y}_i^*)\right\|^2 \quad (28)$$

$$F_i(\mathbf{y}_i) - F_i(\mathbf{y}_i^*) \geq \frac{1}{2L}\|\nabla F_i(\mathbf{y}_i)\|^2. \quad (29)$$

■

Lemma 3: Anticipated average difference between $\overline{\mathbf{y}}^{(t)}$ and $\mathbf{y}_i^{(t)}$ for $i \in B^{(t)}$, where $\overline{\mathbf{y}}^{(t)}$ denotes the current global model

$$\frac{1}{[B]}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left\|\overline{\mathbf{y}}^{(t)} - \mathbf{y}_i^{(t)}\right\|^2\right] \leq 16\eta_t^2\tau^2G^2. \quad (30)$$

Proof:

$$\frac{1}{B}\sum_{i\in B^{(t)}}\left\|\overline{\mathbf{y}}^{(t)} - \mathbf{y}_i^{(t)}\right\|^2 = \frac{1}{B}\sum_{i\in B^{(t)}}\left\|\frac{1}{B}\sum_{i'\in B^{(t)}}\left(\mathbf{y}_{i'}^{(t)} - \mathbf{y}_i^{(t)}\right)\right\|^2 \quad (31)$$

$$\leq \frac{1}{B^2}\sum_{i\in B^{(t)}}\sum_{i'\in B^{(t)}}\left\|\mathbf{y}_{i'}^{(t)} - \mathbf{y}_i^{(t)}\right\|^2 \quad (32)$$

$$= \frac{1}{B^2}\sum_{i\neq i', i,i'\in B^{(t)}}\left\|\mathbf{y}_{i'}^{(t)} - \mathbf{y}_i^{(t)}\right\|^2. \quad (33)$$

According to the update rule, i, i' belong to same attacker set, $B^{(t)}$ and due to that in (33), the summation of $i = i'$ will be zero as in (33). In addition, there exists a t_0 such that $0 \leq t - t_0 < \iota$ for each random t that $\mathbf{y}_{i'}^{(t_0)} = \mathbf{y}_i^{(t_0)}$ thus the global model is upgraded for the selected clients every ι. Therefore, even for a random t, the difference between $\mathbf{y}_{i'}^{(t)}$ and $\mathbf{y}_i^{(t)}$ is upper bounded with ι updates. With not growing η_t over t and $\eta_{t_0} \leq 2\eta_t$, (33) can be further narrowed as

$$\frac{1}{B^2}\sum_{i\neq i', i,i'\in B^{(t)}}\left\|\mathbf{y}_{i'}^{(t)} - \mathbf{y}_i^{(t)}\right\|^2 \quad (34)$$

$$\leq \frac{1}{B^2}\sum_{\substack{i\neq i',\\ i,i'\in B^{(t)}}}\left\|\sum_{j=t_0}^{t_0+\iota-1}\eta_j\left(g_{i'}\left(\mathbf{y}_{i'}^{(j)}, \xi_{i'}^{(j)}\right) - g_i\left(\mathbf{y}_i^{(j)}, \xi_i^{(j)}\right)\right)\right\|^2. \quad (35)$$

$$\leq \frac{\eta_{t_0}^2\iota}{B^2}\sum_{i\neq i',}\sum_{i,i'\in B^{(t)}}^{t_0+\iota-1}\left\|\left(g_{i'}\left(\mathbf{y}_{i'}^{(j)}, \xi_{i'}^{(j)}\right) - g_i\left(\mathbf{y}_i^{(j)}, \xi_i^{(j)}\right)\right)\right\|^2 \quad (36)$$

$$\leq \frac{\eta_{t_0}^2\iota}{B^2}\sum_{i\neq i',}\sum_{j=t_0}^{t_0+\iota-1}\left[2\left\|g_{i'}\left(\mathbf{y}_{i'}^{(j)}, \xi_{i'}^{(j)}\right)\right\|^2 + 2\left\|g_i\left(\mathbf{y}_i^{(j)}, \xi_i^{(j)}\right)\right\|^2\right]. \quad (37)$$

In terms of expectation over (37)

$$\mathbb{E}\left[\frac{1}{B^2}\sum_{i\neq i', i,i'\in B^{(t)}}\left\|\mathbf{y}_{i'}^{(t)} - \mathbf{y}_i^{(t)}\right\|^2\right] \quad (38)$$

$$\leq \frac{2\eta_{t_0}^2\iota}{B^2}\mathbb{E}\left[\sum_{\substack{i\neq i',\\ i,i'\in B^{(t)}}}\sum_{j=t_0}^{t_0+\iota-1}\left(\left\|g_{i'}\left(\mathbf{y}_{i'}^{(j)}, \xi_{i'}^{(j)}\right)\right\|^2 + \left\|g_i\left(\mathbf{y}_i^{(j)}, \xi_i^{(j)}\right)\right\|^2\right)\right] \quad (39)$$

$$\leq \frac{2\eta_{t_0}^2\iota}{B^2}\mathbb{E}_{B^{(t)}}\left[\sum_{i\neq i', i_j}\sum_{i,i'\in B^{(t)}}^{t_0+\iota-1}2G^2\right] \quad (40)$$

$$= \frac{2\eta_{t_0}^2\iota}{B^2}\mathbb{E}_{B^{(t)}}\left[\sum_{\substack{i\neq i'\\ i,i'\in B^{(t)}}}2\iota G^2\right] \quad (41)$$

$$\leq \frac{16\eta_t^2(B-1)\iota^2G^2}{B} \quad (42)$$

$$\leq 16\eta_t^2\iota^2G^2 \quad (43)$$

■

where (42) given that there are no more than $B(B\ 1)$ possible pairs such that $i \neq i'$ in $B^{(t)}$.

Lemma 4: In terms expectation over $\|\overline{\mathbf{y}}^{(t)} - \mathbf{y}^*\|^2$ for FedXPro

$$\mathbb{E}\left[\left\|\overline{\mathbf{y}}^{(t)} - \mathbf{y}^*\right\|^2\right] \leq \frac{1}{B}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left\|\mathbf{y}_i^{(t)} - \mathbf{y}^*\right\|^2\right]. \quad (44)$$

Proof:

$$\mathbb{E}\left[\left\|\overline{\mathbf{y}}^{(t)} - \mathbf{y}^*\right\|^2\right] \quad (45)$$

$$= \mathbb{E}\left[\left\|\frac{1}{B}\sum_{i\in B^{(t)}}\mathbf{y}_i^{(t)} - \mathbf{y}^*\right\|^2\right] = \mathbb{E}\left[\left\|\frac{1}{B}\sum_{i\in B^{(t)}}\left(\mathbf{y}_i^{(t)} - \mathbf{y}^*\right)\right\|^2\right] \quad (46)$$

$$\leq \frac{1}{B}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left\|\mathbf{y}_i^{(t)} - \mathbf{y}^*\right\|^2\right]. \quad (47)$$

■

APPENDIX C
PROOF OF THEOREM 1

Proof: As described in Section II-C, with $\overline{\mathbf{g}}^{(t)} = (1/B)\sum_{i\in B^{(t)}} g_i(\mathbf{y}_i^{(t)}, \xi_i^{(t)})$ and we get that

$$\left\|\overline{\mathbf{y}}^{(t+1)} - \mathbf{y}^*\right\|^2 = \left\|\overline{\mathbf{y}}^{(t)} - \eta_t\overline{\mathbf{g}}^{(t)} - \mathbf{y}^*\right\|^2 \quad (48)$$

$$= \left\|\overline{\mathbf{y}}^{(t)} - \eta_t\overline{\mathbf{g}}^{(t)} - \mathbf{y}^* - \frac{\eta_t}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right) + \frac{\eta_t}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\|^2 \quad (49)$$

$$= \left\|\overline{\mathbf{y}}^{(t)} - \mathbf{y}^* - \frac{\eta_t}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\|^2 + \eta_t^2\left\|\frac{1}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right) - \overline{\mathbf{g}}^{(t)}\right\|^2 \quad (50)$$

$$+ 2\eta_t\left\langle\overline{\mathbf{y}}^{(t)} - \mathbf{y}^* - \frac{\eta_t}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right), \frac{1}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right) - \overline{\mathbf{g}}^{(t)}\right\rangle \quad (51)$$

$$= \left\|\overline{\mathbf{y}}^{(t)} - \mathbf{y}^*\right\|^2 \underbrace{-2\eta_t\left\langle\overline{\mathbf{y}}^{(t)} - \mathbf{y}^*, \frac{1}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\rangle}_{A_1}$$

$$+\underbrace{2\eta_t\left\langle \bar{\mathbf{y}}^{(t)}-\mathbf{y}^*-\frac{\eta_t}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right), \frac{1}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right)-\bar{\mathbf{g}}^{(t)}\right\rangle}_{A_2}$$

$$+\underbrace{\eta_t^2\left\|\frac{1}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\|^2}_{A_3}+\underbrace{\eta_t^2\left\|\frac{1}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right)-\bar{\mathbf{g}}^{(t)}\right\|^2}_{A_4}. \quad (52)$$

Begin by bounding A_1

$$-2\eta_t\left\langle \bar{\mathbf{y}}^{(t)}-\mathbf{y}^*, \frac{1}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\rangle$$

$$=-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left\langle \bar{\mathbf{y}}^{(t)}-\mathbf{y}^*, \nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\rangle \quad (53)$$

$$=-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left\langle \bar{\mathbf{y}}^{(t)}-\mathbf{y}_i^{(t)}, \nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\rangle$$
$$-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left\langle \mathbf{y}_i^{(t)}-\mathbf{y}^*, \nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\rangle \quad (54)$$

$$\le\frac{\eta_t}{B}\sum_{i\in B^{(t)}}\left(\frac{1}{\eta_t}\left\|\bar{\mathbf{y}}^{(t)}-\mathbf{y}_k^{(t)}\right\|^2+\eta_t\left\|\nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\|^2\right)$$
$$-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left\langle \mathbf{y}_i^{(t)}-\mathbf{y}^*, \nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\rangle \quad (55)$$

$$=\frac{1}{B}\sum_{i\in B^{(t)}}\left\|\bar{\mathbf{y}}^{(t)}-\mathbf{y}_i^{(t)}\right\|^2+\frac{\eta_t^2}{B}\sum_{i\in B^{(t)}}\left\|\nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\|^2$$
$$-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left\langle \mathbf{y}_i^{(t)}-\mathbf{y}^*, \nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\rangle \quad (56)$$

$$\le\frac{1}{B}\sum_{i\in B^{(t)}}\left\|\bar{\mathbf{y}}^{(t)}-\mathbf{y}_i^{(t)}\right\|^2+\frac{2L\eta_t^2}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i^*\right)$$
$$-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left\langle \mathbf{y}_i^{(t)}-\mathbf{y}^*, \nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\rangle \quad (57)$$

$$\le\frac{1}{B}\sum_{i\in B^{(t)}}\left\|\bar{\mathbf{y}}^{(t)}-\mathbf{y}_i^{(t)}\right\|^2+\frac{2L\eta_t^2}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i^*\right)$$
$$-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left[\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i(\mathbf{y}^*)\right)+\frac{\mu}{2}\left\|\mathbf{y}_i^{(t)}-\mathbf{y}^*\right\|^2\right] \quad (58)$$

$$\le 16\eta_t^2\iota^2G^2-\frac{\eta_t\mu}{B}\sum_{i\in B^{(t)}}\left\|\mathbf{y}_i^{(t)}-\mathbf{y}^*\right\|^2$$
$$+\frac{2L\eta_t^2}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i^*\right)$$
$$-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i(\mathbf{y}^*)\right) \quad (59)$$

where AM-GM and Cauchy–Schwarz inequalities bound with (54), Lemma 1 bound with (56) due to the μ-convexity of F_i, and Lemma 2 leads to (58). Due to the unbiased gradient, $\mathbb{E}[A_2]=0$. It bounds A_3 by Lemma 3 as follows:

$$\eta_t^2\left\|\frac{1}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\|^2=\frac{\eta_t^2}{B}\sum_{i\in B^{(t)}}\left\|\nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\|^2 \quad (60)$$
$$\le\frac{2L\eta_t^2}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i^*\right). \quad (61)$$

Finally, A_4 can be bounded utilizing the SGD variance bound

$$\mathbb{E}\left[\eta_t^2\left\|\frac{1}{B}\sum_{i\in B^{(t)}}\nabla F_i\left(\mathbf{y}_i^{(t)}\right)-\bar{\mathbf{g}}^{(t)}\right\|^2\right]$$
$$=\eta_t^2\mathbb{E}\left[\left\|\sum_{i\in B^{(t)}}\frac{1}{B}\left(g_i\left(\mathbf{y}_i^{(t)},\xi_i^{(t)}\right)-\nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right)\right\|^2\right] \quad (62)$$
$$=\frac{\eta_t^2}{i^2}\mathbb{E}_{B^{(t)}}\left[\sum_{i\in B^{(t)}}\mathbb{E}\left\|g_i\left(\mathbf{y}_i^{(t)},\xi_i^{(t)}\right)-\nabla F_i\left(\mathbf{y}_i^{(t)}\right)\right\|^2\right] \quad (63)$$
$$\le\frac{\eta_t^2\sigma^2}{i}. \quad (64)$$

By utilizing the previously established bounds of $A_1, A_2, A_3,$ and A_4 we can bound the expectation of the LHS of (48)

$$\mathbb{E}\left[\left\|\bar{\mathbf{y}}^{(t+1)}-\mathbf{y}^*\right\|^2\right]\le\mathbb{E}\left[\left\|\bar{\mathbf{y}}^{(t)}-\mathbf{y}^*\right\|^2\right]$$
$$-\frac{\eta_t\mu}{m}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left\|\mathbf{y}_i^{(t)}-\mathbf{y}^*\right\|^2\right]+16\eta_t^2\iota^2G^2$$
$$+\frac{\eta_t^2\sigma^2}{m}+\frac{4L\eta_t^2}{B}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i^*\right)\right] \quad (65)$$
$$-\frac{2\eta_t}{B}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i(\mathbf{y}^*)\right)\right] \quad (66)$$
$$\le(1-\eta_t\mu)\mathbb{E}\left[\left\|\bar{\mathbf{y}}^{(t)}-\mathbf{y}^*\right\|^2\right]+16\eta_t^2\iota^2G^2$$
$$+\frac{\eta_t^2\sigma^2}{B} \quad (67)$$
$$+\underbrace{\frac{4L\eta_t^2}{B}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i^*\right)\right]-\frac{2\eta_t}{B}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i(\mathbf{y}^*)\right)\right]}_{A_5}. \quad (68)$$

Lemma 3 leads to (68). In terms of bounding A_5 in (68), we can show A_5 in a alternative format as

$$\mathbb{E}\left[\frac{4L\eta_t^2}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i^*\right)-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i(\mathbf{y}^*)\right)\right]$$
$$=\mathbb{E}\left[\frac{4L\eta_t^2}{B}\sum_{k\in B^{(t)}}F_i\left(\mathbf{y}_i^{(t)}\right)-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}F_i\left(\mathbf{y}_i^{(t)}\right)\right]$$
$$-\left[\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left(F_i^*-F_i(\mathbf{y}^*)\right)\right]$$
$$+\left[\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}F_i^*-\frac{4L\eta_t^2}{B}\sum_{i\in B^{(t)}}F_i^*\right]$$

$$= \underbrace{\mathbb{E}\left[\frac{2\eta_t(2L\eta_t-1)}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i^*\right)\right]}_{A_6} + 2\eta_t\mathbb{E}\left[\frac{1}{B}\sum_{i\in B^{(t)}}\left(F_i(\mathbf{y}^*)-F_i^*\right)\right]. \quad (69)$$

Now with $\eta_t < 1/(4L)$ and $\nu_t = 2\eta_t(1-2L\eta_t)$, we can rearrange the A_6 and later bound the A_6 as

$$-\frac{\nu_t}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i\left(\bar{\mathbf{y}}^{(t)}\right)+F_i\left(\bar{\mathbf{y}}^{(t)}\right)-F_i^*\right)$$

$$= -\frac{\nu_t}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i\left(\bar{\mathbf{y}}^{(t)}\right)\right)-\frac{\nu_t}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\bar{\mathbf{y}}^{(t)}\right)-F_i^*\right) \quad (70)$$

$$\leq -\frac{\nu_t}{B}\sum_{i\in B^{(t)}}\left[\left\langle\nabla F_i\left(\bar{\mathbf{y}}^{(t)}\right),\mathbf{y}_k^{(t)}-\bar{\mathbf{y}}^{(t)}\right\rangle+\frac{\mu}{2}\left\|\mathbf{y}_i^{(t)}-\bar{\mathbf{y}}^{(t)}\right\|^2\right] - \frac{\nu_t}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\bar{\mathbf{y}}^{(t)}\right)-F_i^*\right) \quad (71)$$

$$\leq \frac{\nu_t}{B}\sum_{i\in B^{(t)}}\left[\eta_t L\left(F_i\left(\bar{\mathbf{y}}^{(t)}\right)-F_i^*\right)+\left(\frac{1}{2\eta_t}-\frac{\mu}{2}\right)\left\|\mathbf{y}_i^{(t)}-\bar{\mathbf{y}}^{(t)}\right\|^2\right] - \frac{\nu_t}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\bar{\mathbf{y}}^{(t)}\right)-F_i^*\right) \quad (72)$$

$$= -\frac{\nu_t}{B}(1-\eta_t L)\sum_{i\in B^{(t)}}\left(F_i\left(\bar{\mathbf{y}}^{(t)}\right)-F_i^*\right) + \left(\frac{\nu_t}{2\eta_t B}-\frac{\nu_t\mu}{2B}\right)\sum_{i\in B^{(t)}}\left\|\mathbf{y}_k^{(t)}-\bar{\mathbf{y}}^{(t)}\right\|^2 \quad (73)$$

$$\leq -\frac{\nu_t}{B}(1-\eta_t L)\sum_{i\in B^{(t)}}\left(F_i\left(\bar{\mathbf{y}}^{(t)}\right)-F_i^*\right)+\frac{1}{B}\sum_{i\in B^{(t)}}\left\|\mathbf{y}_i^{(t)}-\bar{\mathbf{y}}^{(t)}\right\|^2. \quad (74)$$

While (71) results from the μ-convexity, (72) from Lemma 1 and the AM-GM and the Cauchy–Schwarz inequalities, and (74) from the fact that fact that $([\nu\nu_t(1-\eta_t\mu)]/2\eta_t) \leq 1$. Hence, we can upper bound A_5 utilizing bound of A_6

$$\mathbb{E}\left[\frac{4L\eta_t^2}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i^*\right)-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i(\mathbf{y}^*)\right)\right]$$

$$\leq \frac{1}{B}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left\|\mathbf{y}_i^{(t)}-\bar{\mathbf{y}}^{(t)}\right\|^2\right] - \frac{\nu_t}{B}(1-\eta_t L)\mathbb{E}\left[\sum_{i\in B^{(t)}}\left(F_i\left(\bar{\mathbf{y}}^{(t)}\right)-F_i^*\right)\right] + \frac{2\eta_t}{B}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left(F_i(\mathbf{y}^*)-F_i^*\right)\right] \quad (75)$$

$$\leq 16\eta_t^2\iota^2G^2-\frac{\nu_t}{B}(1-\eta_t L)\mathbb{E}\left[\sum_{i\in B^{(t)}}\left(F_i\left(\bar{\mathbf{y}}^{(t)}\right)-F_k^*\right)\right] + \frac{2\eta_t}{B}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left(F_i(\mathbf{y}^*)-F_i^*\right)\right] \quad (76)$$

$$= 16\eta_t^2\iota^2G^2 - \nu_t(1-\eta_t L)\mathbb{E}\left[f\left(B\left(\Theta,\bar{\mathbf{y}}^{(\iota\lfloor t/\iota\rfloor)}\right),\bar{\mathbf{y}}^{(t)}\right)\left(F\left(\bar{\mathbf{y}}^{(t)}\right)-\sum_{i=1}^{i}p_iF_i^*\right)\right] + 2\eta_t\mathbb{E}\left[f\left(B\left(\Theta,\bar{\mathbf{y}}^{(\iota\lfloor t/\iota\rfloor)}\right),\mathbf{y}^*\right)\left(F^*-\sum_{k=1}^{i}\alpha_iF_i^*\right)\right] \quad (77)$$

$$\leq 16\eta_t^2\iota^2G^2\underbrace{-\nu_t(1-\eta_t L)\bar{f}\left(\mathbb{E}\left[F\left(\bar{\mathbf{y}}^{(t)}\right)\right]-\sum_{i=1}^{i}p_iF_i^*\right)}_{A_7}+2\eta_t\tilde{f}\hbar \quad (78)$$

where, $f(B(\Theta,\mathbf{y}),\mathbf{y}')$ in Definition 2 leads to (77) and Definition 1 leads to (78). Since we define $\bar{f}$ and $\tilde{f}$ in Definition 2, A_7 in (78) can be expanded as

$$= -\nu_t(1-\eta_t L)\bar{f}\left(\mathbb{E}\left[F\left(\bar{\mathbf{y}}^{(t)}\right)\right]-\sum_{i=1}^{i}p_iF_i^*\right) \quad (79)$$

$$= -\nu_t(1-\eta_t L)\bar{f}\sum_{i=1}^{i}p_i\left(\mathbb{E}\left[F_i\left(\bar{\mathbf{y}}^{(t)}\right)\right]-F^*+F^*-F_i^*\right) \quad (80)$$

$$= -\nu_t(1-\eta_t L)\bar{f}\sum_{i=1}^{i}p_i\left(\mathbb{E}\left[F_i\left(\bar{\mathbf{y}}^{(t)}\right)\right]-F^*\right) - \nu_t(1-\eta_t L)\bar{f}\sum_{i=1}^{i}p_i\left(F^*-F_i^*\right) \quad (81)$$

$$= -\nu_t(1-\eta_t L)\bar{f}\left(\mathbb{E}\left[F\left(\bar{\mathbf{y}}^{(t)}\right)\right]-F^*\right)-\nu_t(1-\eta_t L)\bar{f}\hbar \quad (82)$$

$$\leq -\frac{\nu_t(1-\eta_t L)\mu\bar{f}}{2}\mathbb{E}\left[\left\|\bar{\mathbf{y}}^{(t)}-\mathbf{y}^*\right\|^2\right]-\nu_t(1-\eta_t L)\bar{f}\hbar \quad (83)$$

$$\leq -\frac{3\eta_t\mu\bar{f}_8}{8}\left[\left\|\bar{\mathbf{y}}^{(t)}-\mathbf{y}^*\right\|^2\right]-2\eta_t(1-2L\eta_t)(1-\eta_t L)\bar{f}\hbar \quad (84)$$

$$\leq -\frac{3\eta_t\mu\bar{f}}{8}\mathbb{E}\left[\left\|\bar{\mathbf{y}}^{(t)}-\mathbf{y}^*\right\|^2\right]-2\eta_t\bar{f}\hbar+6\eta_t^2\bar{f}L\hbar. \quad (85)$$

where μ-convexity accounts for (83), condition $-2\eta_t(1-2L\eta_t)(1-\eta_t L) \leq -(3/4)\eta_t$ accounts for (84), and $(1-2L\eta_t)(1-\eta_t L) \leq -(1-3L\eta_t)$ accounts for (85). Hence, A_5 can be finally bounded as

$$\frac{4L\eta_t^2}{B}\mathbb{E}\left[\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_k^*\right)-\frac{2\eta_t}{B}\sum_{i\in B^{(t)}}\left(F_i\left(\mathbf{y}_i^{(t)}\right)-F_i(\mathbf{y}^*)\right)\right]$$

$$\leq -\frac{3\eta_t\mu\bar{f}^8}{8}\left[\left\|\bar{\mathbf{y}}^{(t)}-\mathbf{y}^*\right\|^2\right]+2\eta_t\hbar\left(\tilde{f}-\bar{f}\right)+\eta_t^2\left(6\bar{f}L\hbar+16\iota^2G^2\right). \quad (86)$$

Now, $\mathbb{E}[\|\bar{\mathbf{y}}^{(t+1)}-\mathbf{y}^*\|^2]$ can be bounded as

$$\mathbb{E}\left[\left\|\bar{\mathbf{y}}^{(t+1)}-\mathbf{y}^*\right\|^2\right] \leq \left[1-\eta_t\mu\left(1+\frac{3\bar{f}}{8}\right)\right]\mathbb{E}\left[\left\|\bar{\mathbf{y}}^{(t)}-\mathbf{y}^*\right\|^2\right] + \eta_t^2\left(32\iota^2G^2+\frac{\sigma^2}{B}+6\bar{f}L\hbar\right)+2\eta_t\hbar\left(\tilde{f}-\bar{f}\right). \quad (87)$$

By specifying $\Delta_{t+1} = \mathbb{E}[\|\bar{\mathbf{y}}^{(t+1)}-\mathbf{y}^*\|^2]$, $M = 1+(3\bar{f}/8)$, $C = 32\iota^2G^2+(\sigma^2/B)+6\bar{f}L\hbar$, $D = 2\hbar(\tilde{f}-\bar{f})$, we have that

$$\Delta_{t+1} \leq (1-\eta_t\mu M)\Delta_t+\eta_t^2C+\eta_t D. \quad (88)$$

Using $\Delta_t \leq (\psi/[t+\gamma])$, $\eta_t = (\beta/[t+\gamma])$ and $\beta > (1/\mu M)$, $\gamma > 0$ by induction we have that

$$\psi = \max\left\{\gamma\left\|\bar{\mathbf{y}}^{(0)}-\mathbf{y}^*\right\|^2, \frac{1}{\beta\mu M-1}\left(\beta^2C+D\beta(t+\gamma)\right)\right\}. \quad (89)$$

Taking into account the L-smoothness of $F(\cdot)$, we then get

$$\mathbb{E}\Big[F\Big(\bar{\mathbf{y}}^{(t)}\Big)\Big] - F^* \le \frac{L}{2}\Delta_t \le \frac{L}{2}\frac{\psi}{\gamma + t}. \tag{90}$$

References

[1] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, “Evaluating critical security issues of the IoT world: Present and future challenges,” *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2483–2495, Aug. 2018.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[3] T. Gafni, N. Shlezinger, K. Cohen, Y. C. Eldar, and H. V. Poor, “Federated learning: A signal processing perspective,” *IEEE Signal Process. Mag.*, vol. 39, no. 3, pp. 14–41, May 2022.

[4] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, “Federated learning for Internet of Things: A comprehensive survey,” *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1622–1658, 3rd Quart., 2021.

[5] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Y. Zomaya, “An efficiency-boosting client selection scheme for federated learning with fairness guarantee,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1552–1564, Jul. 2021.

[6] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, and A. Y. Zomaya, “Federated learning for COVID-19 detection with generative adversarial networks in edge cloud computing,” *IEEE Internet Things J.*, vol. 9, no. 12, pp. 10257–10271, Jun. 2022.

[7] D. C. Nguyen et al., “Federated learning meets blockchain in edge computing: Opportunities and challenges,” *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12806–12825, Aug. 2021.

[8] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, “Communication-efficient federated learning,” *Proc. Nat. Acad. Sci. USA*, vol. 118, no. 17, p. 19, 2021.

[9] N. Bouacida and P. Mohapatra, “Vulnerabilities in federated learning,” *IEEE Access*, vol. 9, pp. 63229–63249, 2021.

[10] J. Men et al., “Finding sands in the eyes: Vulnerabilities discovery in IoT with EUFuzzer on human machine interface,” *IEEE Access*, vol. 7, pp. 103751–103759, 2019.

[11] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, “Machine learning with adversaries: Byzantine tolerant gradient descent,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, p. 19.

[12] Y. Chen, L. Su, and J. Xu, “Distributed statistical machine learning in adversarial settings: Byzantine gradient descent,” in *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, 2017, pp. 1–25.

[13] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, “Byzantine-robust distributed learning: Towards optimal statistical rates,” in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.

[14] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, “How to backdoor federated learning,” in *Proc. Int. Conf. Artif. Intell. Stat.*, 2020, pp. 2938–2948.

[15] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, “Analyzing federated learning through an adversarial lens,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 634–643.

[16] J. Xu, S.-L. Huang, L. Song, and T. Lan, “SignGuard: Byzantine-robust federated learning through collaborative malicious gradient filtering,” 2021, *arXiv:2109.05872*.

[17] X. Cao, J. Jia, and N. Z. Gong, “Provably secure federated learning against malicious clients,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 6885–6893.

[18] V. Shejwalkar and A. Houmansadr, “Manipulating the Byzantine: Optimizing model poisoning attacks and defenses for federated learning,” in *Proc. NDSS*, 2021, pp. 1–21.

[19] L. Nagalapatti and R. Narayanam, “Game of gradients: Mitigating irrelevant clients in federated learning,” 2021, *arXiv:2110.12257*.

[20] K. Zhao, W. Xi, Z. Wang, J. Zhao, R. Wang, and Z. Jiang, “SMSS: Secure member selection strategy in federated learning,” *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 37–49, Jul./Aug. 2020.

[21] X. Cao, M. Fang, J. Liu, and N. Z. Gong, “FLTrust: Byzantine-robust federated learning via trust bootstrapping,” 2020, *arXiv:2012.13995*.

[22] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, “Federated variance-reduced stochastic gradient descent with robustness to Byzantine attacks,” *IEEE Trans. Signal Process.*, vol. 68, pp. 4583–4596, 2020.

[23] Z. Gu and Y. Yang, “Detecting malicious model updates from federated learning on conditional variational autoencoder,” in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2021, pp. 671–680.

[24] N. Rodríguez-Barroso, E. Martínez-Cámara, M. Luzón, G. G. Seco, M. Á. Veganzones, and F. Herrera, “Dynamic federated learning model for identifying adversarial clients,” 2020, *arXiv:2007.15030*.

[25] S. Shen, S. Tople, and P. Saxena, “AUROR: Defending against poisoning attacks in collaborative deep learning systems,” in *Proc. 32nd Annu. Conf. Comput. Security Appl.*, 2016, pp. 508–519.

[26] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, “RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 1544–1551.

[27] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, “Abnormal client behavior detection in federated learning,” 2019, *arXiv:1910.09933*.

[28] J. Lin, M. Du, and J. Liu, “Free-riders in federated learning: Attacks and defenses,” 2019, *arXiv:1911.12560*.

[29] M. Fang, X. Cao, J. Jia, and N. Gong, “Local model poisoning attacks to {Byzantine-robust} federated learning,” in *Proc. 29th USENIX Security Symp. (USENIX Security)*, 2020, pp. 1605–1622.

[30] S. Hu, J. Lu, W. Wan, and L. Y. Zhang, “Challenges and approaches for mitigating Byzantine attacks in federated learning,” 2021, *arXiv:2112.14468*.

[31] W. Wan, J. Lu, S. Hu, L. Y. Zhang, and X. Pei, “Shielding federated learning: A new attack approach and its defense,” in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2021, pp. 1–7.

[32] M. W. Spratling, “A neural implementation of Bayesian inference based on predictive coding,” *Connection Sci.*, vol. 28, no. 4, pp. 346–383, 2016.

[33] L. Lamport, R. Shostak, and M. Pease, “The Byzantine generals problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[34] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, “Adversarial machine learning,” in *Proc. 4th ACM Workshop Security Artif. Intell.*, 2011, pp. 43–58.

[35] C. Fung, C. J. Yoon, and I. Beschastnikh, “Mitigating Sybils in federated learning poisoning,” 2018, *arXiv:1808.04866*.

[36] K. Aftab, R. Hartley, and J. Trumpf, “Generalized Weiszfeld algorithms for LQ optimization,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 4, pp. 728–745, Apr. 2015.

[37] Y. J. Cho, J. Wang, and G. Joshi, “Client selection in federated learning: Convergence analysis and power-of-choice selection strategies,” 2020, *arXiv:2010.01243*.

[38] N. Kumar and M. Nachamai, “Noise removal and filtering techniques used in medical images,” *Orient J. Comput. Sci. Technol.*, vol. 10, no. 1, p. 16, 2017.

[39] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” 2017, *arXiv:1706.06083*.

[40] X. Li, Z. Qu, S. Zhao, B. Tang, Z. Lu, and Y. Liu, “LoMar: A local defense against poisoning attack on federated learning,” *IEEE Trans. Depend. Secure Comput.*, vol. 20, no. 1, pp. 437–450, Jan./Feb. 2023.

Pubudu L. Indrasiri (Member, IEEE) received the B.Sc. degree in computer science from the University of Colombo, Colombo, Sri Lanka, in 2016, and the M.Sc. degree in network security from Charles Sturt University, Melbourne, VIC, Australia, in 2020. He is currently pursuing the Ph.D. degree in artificial intelligence with Deakin University, Geelong, VIC, Australia.

His research interests include machine learning, deep learning, decentralized deep learning, building performance enhanced deep learning architectures, federated learning, and the integration of privacy and security techniques into these systems.

Dinh C. Nguyen (Member, IEEE) received the Ph.D. degree from Deakin University, Geelong, VIC, Australia, in 2021.

He is an Assistant Professor with the Department of Electrical and Computer Engineering, The University of Alabama at Huntsville, Huntsville, AL, USA. He was a Postdoctoral Research Associate with Purdue University, West Lafayette, IN, USA, from 2022 to 2023. He has published over 40 papers on top-tier IEEE/ACM conferences and journals. His research interests include wireless networking, distributed learning, security, and privacy.

Dr. Nguyen is an Associate Editor of the IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY.

Bipasha Kashyap (Member, IEEE) received the B.E. degree (First-Class Hons.) in electronics and telecommunication engineering from Gauhati University, Guwahati, India, in 2011, the M.E. degree (First-Class Hons.) in electronics and management engineering from Deakin University, Geelong, VIC, Australia, in 2016, and the Ph.D. degree in biomedical engineering from Deakin University in 2021, being honored with the Data61 Ph.D. Scholarship by CSIRO, Canberra, ACT, Australia.

She is currently a Lecturer with the School of Engineering, Deakin University. From 2012 to 2014, she served as a Software Engineer with Tech Mahindra, Hyderabad, India. Her research interests encompass cognitive computational neuroscience, acoustic signal processing, human motion analytics, scalable machine learning algorithms, and AI/ML pipeline development in cloud and network engineering.

Dr. Kashyap has also been a Distinguished Recipient of the prestigious Veski Inspiring Women Victorian Fellowship in Australia, which led to her appointment as a Hardware Engineer with the Australian Road Research Board in 2016.

Pubudu N. Pathirana (Senior Member, IEEE) was born in Matara, Sri Lanka, in 1970. He was educated with Royal College, Colombo, Sri Lanka. He received the B.E. degree (Hons.) in electrical engineering and the B.Sc. degree in mathematics, in 1996, and the Ph.D. degree in electrical engineering from The University of Western Australia, Perth, WA, Australia, in 2000, all sponsored by the Government of Australia on EMSS and IPRS Scholarships, respectively.

He was a Postdoctoral Research Fellow of the University of Oxford, Oxford, U.K.; a Research Fellow of the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW, Australia; and a Consultant with the Defence Science and Technology Organization, Fairbairn, CA, Australia, in 2002. He was a Visiting Professor with Yale University, New Haven, CT, USA, in 2009. He is currently a Full Professor, the Head of Discipline, Mechatronics, Electrical, and Electronic Engineering, and the Director of the Networked Sensing and Control Research Group, School of Engineering, Deakin University, Geelong, VIC, Australia. His current research interests include bio-medical assistive device design, human motion capture, mobile/wireless and the IoT networks, rehabilitation robotics, and signal processing.

Yonina C. Eldar (Fellow, IEEE) received the first B.Sc. degree in physics and the second B.Sc. degree in electrical engineering from Tel-Aviv University, Tel Aviv, Israel, in 1995 and 1996, respectively, and the Ph.D. degree in electrical engineering and computer science from Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2002.

She is a Professor with the Department of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel, where she heads the Center for Biomedical Engineering and Signal Processing and holds the Dorothy and Patrick Gorman Professorial Chair. She is also a Visiting Professor with MIT, a Visiting Scientist with the Broad Institute, and an Adjunct Professor with Duke University, Durham, NC, USA, and was a Visiting Professor with Stanford University, Stanford, CA, USA.

Prof. Eldar has received many awards for excellence in research and teaching, including the IEEE Signal Processing Society Technical Achievement Award in 2013, the IEEE/AESS Fred Nathanson Memorial Radar Award in 2014, and the IEEE Kiyo Tomiyasu Award in 2016. She was a Horev Fellow of the Leaders in Science and Technology Program at the Technion and an Alon Fellow. She received the Michael Bruno Memorial Award from the Rothschild Foundation, the Weizmann Prize for Exact Sciences, the Wolf Foundation Krill Prize for Excellence in Scientific Research, the Henry Taub Prize for Excellence in Research (twice), the Hershel Rich Innovation Award (three times), and the Award for Women with Distinguished Contributions. She received several best paper awards and best demo awards together with her research students and colleagues, was selected as one of the 50 most influential women in Israel, and was a member of the Israel Committee for Higher Education. She is the Editor-in-Chief of Foundations and Trends in *Signal Processing*, a member of several IEEE Technical Committees and Award Committees, and heads the Committee for Promoting Gender Fairness in Higher Education Institutions in Israel. She is a member of the Israel Academy of Sciences and Humanities and an EURASIP Fellow.