Federated Learning with Unreliable Clients: Performance Analysis and Mechanism Design

Chuan Ma, *Member, IEEE,* Jun Li, *Senior Member, IEEE,*
Ming Ding, *Senior Member, IEEE,* Kang Wei, *Student Member, IEEE,*
Wen Chen, *Senior Member, IEEE,* and H. Vincent Poor, *Life Fellow, IEEE*

Abstract—Owing to the low communication costs and privacy-promoting capabilities, Federated Learning (FL) has become a promising tool for training effective machine learning models among distributed clients. However, with the distributed architecture, low quality models could be uploaded to the aggregator server by unreliable clients, leading to a degradation or even a collapse of training. In this paper, we model these unreliable behaviors of clients and propose a defensive mechanism to mitigate such a security risk. Specifically, we first investigate the impact on the models caused by unreliable clients by deriving a convergence upper bound on the loss function based on the gradient descent updates. Our theoretical bounds reveal that with a fixed amount of total computational resources, there exists an optimal number of local training iterations in terms of convergence performance. We further design a novel defensive mechanism, named deep neural network based secure aggregation (DeepSA). Our experimental results validate our theoretical analysis. In addition, the effectiveness of DeepSA is verified by comparing with other state-of-the-art defensive mechanisms.

Index Terms—Federated learning, Unreliable clients, Convergence bound, Defensive mechanism

I. INTRODUCTION

Machine learning (ML) technologies, e.g., deep learning, have revolutionized the ways that information is extracted with ground breaking successes in various areas. Meanwhile, owing to the advent of the Internet of things (IoT), the number of intelligent applications with edge computing, such as smart manufacturing, intelligent transportation and intelligent logistics, is growing exponentially [1]–[5]. As such, the conventional centralized deep learning is no longer capable of efficiently processing the dramatically increased data from the vast IoT devices. To tackle this challenge, distributed learning frameworks have emerged, e.g., federated learning (FL), enabling the decouple of data provision by distributed clients and aggregating ML models at a centralized server [6]–[8]. Through local training and central aggregating iteratively,

Chuan Ma, Jun Li and Kang Wei are with School of Electrical and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China. E-mail: {kang.wei, jun.li, chuan.ma}@njust.edu.cn.

Ming Ding is with Data61, CSIRO, Sydney, Australia. E-mail: ming.ding@data61.csiro.au.

Wen Chen is with the Department of Electronics Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: wenchen@sjtu.edu.cn.

H. Vincent Poor is with Department of Electrical Engineering, Princeton University, NJ, USA. E-mail: poor@princeton.edu.

FL does not require clients to share their sensitive data with the central server, thereby effectively reducing transmission overheads as well as preserving clients' privacy to some extent [9]–[11].

Although the clients' data are not explicitly exposed in the original format, it is still possible for adversaries to infer clients' private information approximately, especially when the architecture of the FL model and its parameters are not completely protected. Moreover, the existence of unreliable clients may further incur security issues in IoT applications. This is because the server in a FL system has no access to the clients' data, nor does it have a full control of the clients' behaviors. As a consequence, a client may deviate from the normal behaviors during the course of FL, which is termed as **unreliable client** in this work. Unreliable behaviors may be caused intentionally, e.g., by a malicious attacker disguised as a normal client, or unintentionally, e.g., by a client with hardware and/or software limitaions/defects in IoT. For example, in smart manufacturing scenarios, engines with sensors that have abnormal traffic and irregular reporting frequency may cause industrial production interruption thus resulting in huge economic losses for factories [12], [13].

Unreliable clients in FL, for example, may manipulate their outputs sent to the server and they can dominate the training process and change the judging boundary of the global model, or make the global model deviate from the optimal solution. To model these clients, the work in [14] proposed that an unreliable client may interfere with the process of FL by applying limited changes to the uploaded model parameters. The work in [15] proposed a model-replacement method that demonstrated its efficacy on poisoning models of standard FL tasks in IoT. In addition, this work also developed and evaluated a generic constrain-and-scale technique that incorporates the evasion of defensive mechanism into the abnormal clients' loss function during training. Therefore, how to design defensive algorithms against abnormal clients in FL becomes crucial. In order to detect abnormal updates in FL, the work of [16] applied the results of client-side cross-validation for reducing the weights of bad updates when performing aggregation, where each update is evaluated over other clients' local data. Similarly, the work in [17] also focused on the weights and they presented a novel aggregation algorithm with the residual-based re-weighting method. The work in [18] considered the existence of unreliable participants and utilized an auxiliary validation data to compute a utility score for each participant to reduce the impact of these unreliable participants, while

the work in [19] directly removed the corresponding model parameters from the training procedure if the accuracy of client is lower than a predefined threshold. The work in [20] proposed a robust aggregation rule, called adaptive federated averaging, which detects and discards malicious or bad local updates based on a hidden Markov model. The work in [21] performed the first systematic study on local model poisoning attacks on FL, in which they formulate attacks as optimization problems and test four different robust FL methods. However, all of these works lack theoretical analysis on the performance of FL systems with the existence of unreliable clients.

It should be noted that the analysis and optimization for the basic FL system has already been investigated in [22]–[25], yet there are no analytical results on the security aspects in a FL system. Therefore, in this work, we conduct analysis in the context of FL with unreliable clients. We first introduce the unreliable model of clients in FL systems and derive the theoretical convergence bounds. Through our theoretical results, we find that there exists an optimal local training iteration that leads to a best system performance within a constraint on total computing resources. Then, we design a novel defensive mechanism, referred to as deep neural network (DNN) based secure aggregation (DeepSA), to efficiently reduce the negative effects caused by unreliable clients.

The major contributions of this paper can be summarized as follows:

- We involve unreliable clients in FL, which model parameters will be scaled down and noised before uploading. Further, we derive the upper bound of loss function in FL systems with a given computational resources. Our theoretical bound reveals that there exists an optimal number of local training epochs to achieve the best convergence performance.
- We propose a novel defensive mechanism, i.e., DNN-based DeepSA, which can detect abnormal models, and then alleviates the negative effects by removing them from the aggregation.
- We conduct extensive experiments on the proposed model with the multi-layer perceptron model and real-life dataset. Our experimental results are shown to be consistent with the theoretical ones. Also, compared with other existing defensive algorithms, the proposed one can improve the FL model performance effectively.

The rest of this paper is organized as follows. Section II introduces background of FL. Section III details the system models. In Section IV we analyzes convergence bounds of FL system with unreliable clients. Section V proposes the DeepSA algorithm to address the unreliable problem, and the experimental results are shown in Section VI. Finally, we conclude the paper in Section VIII. In addition, a summary of notation is listed in Table I.

II. PRELIMINARIES FOR FEDERATED LEARNING

In this section, we will introduce the basic concepts of FL. As a kind of distributed training frameworks [7], FL can promote user privacy by its unique distributed learning mechanism. In FL, all clients share the same learning objective
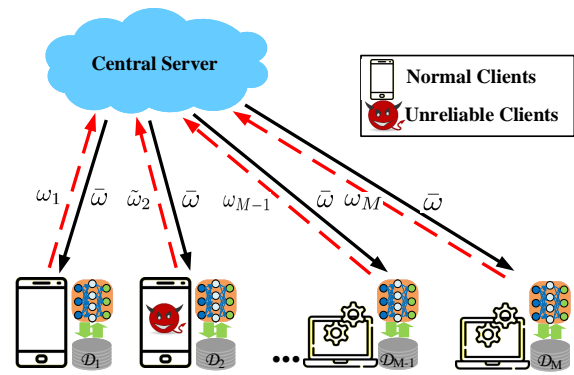

Fig. 1. A FL training framework with unreliable clients.

and model structure, where a central server sends the current global model parameters $\boldsymbol{w}$ to all clients $\mathcal{C}_i$, $\forall i \in \mathcal{M} \triangleq \{1, 2, \ldots, M\}$, in each communication round. Then, all clients update local models based on the shared global model and local data set $\mathcal{D}_i$. After local training, all local models will be uploaded to the server by clients, and then aggregated by the server as the current global model, which is expressed as

$$\boldsymbol{w}^{(k\tau)} = \sum_{i \in \mathcal{M}} p_i \boldsymbol{w}_i^{(k\tau)}, \qquad (1)$$

where $\boldsymbol{w}_i^{(k\tau)}$ is the uploaded model of i-th client at the k-th communication round, $\boldsymbol{w}^{(k\tau)}$ is the global model after aggregation at the k-th communication round, τ is the number of local training epochs, and $p_i = |\mathcal{D}_i|/|\mathcal{D}|$ is the aggregating weight based on the size of $\mathcal{D}_i$, where $\mathcal{D} = \sum_{i \in \mathcal{M}} \cup \mathcal{D}_i$ and $|\cdot|$ represents the cardinality of a set, respectively. At the server side, the goal is to learn a model over data that resides at the M associated clients. Formally, this FL task can be expressed as

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i \in \mathcal{M}} p_i F_i(\boldsymbol{w}), \qquad (2)$$

where $F(\boldsymbol{w}) = \sum_{i \in \mathcal{M}} p_i F_i(\boldsymbol{w})$ and $F_i(\cdot)$ is the local objective function of the i-th client.

In addition, the FL in this paper adopts the optimization method of gradient descent. In order to capture the divergence between the gradient of a local and global loss function, the gradient divergence is defined as follows [24].

Definition 1 *For any $i \in \mathcal{M}$ and $\boldsymbol{w}$, an upper bound of $\|\nabla F_i(\boldsymbol{w}) - \nabla F(\boldsymbol{w})\|$ is defined as δ_i, i.e.,*

$$\|\nabla F_i(\boldsymbol{w}) - \nabla F(\boldsymbol{w})\| \le \delta_i. \qquad (3)$$

We also define $\delta \triangleq \frac{\sum_{i \in \mathcal{M}} |\mathcal{D}_i| \delta_i}{|\mathcal{D}|}$. If the size of each local dataset is same, we know that $\delta = \frac{1}{M} \sum_{i=1}^{M} \delta_i$.

This divergence is governed by how the data is distributed at different clients.

III. SYSTEM MODELS

In FL, the unreliable model updates might exist in a wireless transmission environment. Thus, these flawed uploads will impair the effectiveness of the global model, misleading the

TABLE I
SUMMARY OF MAIN NOTATION

Notation	Description
$\boldsymbol{w}_i^{(k\tau)}$	The uploaded model of i-th client at the k-th communication round
τ	The number of local training epochs
$F_i(\cdot)$	The local objective function of the i-th client
$\widehat{\boldsymbol{w}}_i^{(k\tau)}$	The local model of the i-th unreliable client
α	The scaling factor with range [-1,1]
$\boldsymbol{n}$	The additive noise
p_{U}	The probability of the unreliable behavior
p_i	The aggregating weight based on the training data size of $\mathcal{D}_i$
T	The number of total training iterations
k	The number of total aggregation

updated AI model away from optimality. To be more specific, abnormal behaviors generally can be classified into two categories: intentional and unintentional. Intentional adversary clients, also regarded as malicious clients, usually aim to sabotage the system performance or even destroy the learned model. For example, the values of the uploaded parameters may be scaled down or even completely reversed. By contrast, unintentional behaviors could happen without any particular purpose, for example, a noisy version of parameters could be uploaded to the server. In this paper we propose to model both types of abnormal clients.

We consider a FL system consisting of a single central server and M clients, as shown in Fig. 1. We assume each clients may upload unreliable models throughout the whole training process.

A. Adversary Model

Each client is assumed to have a local model with the same structure, and corresponding model parameters uploaded for each epoch are of the same format. The shared model is guided by these parameter vectors to the optimal value. Then, we denote by $\widehat{\boldsymbol{w}}_i^{(k\tau)}$ the local model of the i-th unreliable client, and express as

$$\widehat{\boldsymbol{w}}_i^{(k\tau)} = \alpha \boldsymbol{w}_i^{(k\tau)} + \boldsymbol{n}_i^{(k\tau)}, \tag{4}$$

where $\alpha \in [-1, 1]$ denotes the scaling factor and $\boldsymbol{n}$ denotes the additive noise and is assumed to follow a Gaussian distribution with $N(0, \sigma^2)$. Eq. (4) can well capture the abnormal behaviors as the scalar is used to model the malicious clients and the random noise denotes the undesirable perturbation on the uploaded models. For example, when $\alpha = -1$, it means an adversarial client will completely reverse the uploaded parameters on purpose, and can be recognized as a malicious behavior. Due to the aggregation process in Eq. (1), we have

$$\bar{\boldsymbol{w}}^{(k\tau)} = \sum_{i \in M} p_i \left[(1 - p_{\mathrm{U}}) \boldsymbol{w}_i^{(k\tau)} + p_{\mathrm{U}} \hat{\boldsymbol{w}}_i^{(k\tau)} \right], \tag{5}$$

where p_{U} denotes the probability of the unreliable behavior[1].

---

[1] We assume a same value of p_{U} for all clients in this work. Different unreliable probabilities for different clients maybe out of the scope, and can be our future work.

IV. CONVERGENCE ANALYSIS

In this section, we will propose a theoretical analysis on the convergence of the FL system considering the existing of abnormal behaviors. For the purpose of facilitating the analysis, we make the following assumptions on the loss function.

Assumption 1 *We assume the following conditions are satisfied for all i, $\forall i \in \mathcal{M}$:*

1) *$F_i(\boldsymbol{w})$ is convex;*
2) *All model parameters satisfy $\|\boldsymbol{w}\| \leq \Theta$;*
3) *$F_i(\boldsymbol{w})$ is ρ-Lipschitz, i.e., $\|F_i(\boldsymbol{w}) - F_i(\boldsymbol{w}')\| \leq \rho\|\boldsymbol{w} - \boldsymbol{w}'\|$, for any $\boldsymbol{w}$, $\boldsymbol{w}'$;*
4) *$F_i(\boldsymbol{w})$ is β-smooth, i.e., $\|\nabla F_i(\boldsymbol{w}) - \nabla F_i(\boldsymbol{w}')\| \leq \beta\|\boldsymbol{w} - \boldsymbol{w}'\|$, for any $\boldsymbol{w}$, $\boldsymbol{w}'$;*
5) *$\eta \leq \frac{1}{\beta}$, where η is the step size;*
6) *$\|F(\boldsymbol{w}^t) - F(\boldsymbol{w}^*)\| \geq \varepsilon$, for all $\boldsymbol{w}$ during FL training.*

We also assume that the clients participating in the training hold the same amount of data, i.e. $p_i = \frac{1}{M}$. In general, these assumptions with some restrictions for the convenience of theoretical derivation can be satisfied.

A. Convergence Analysis

In this subsection, we evaluate the performance of FL under the abnormal behaviors by an upper bound on the difference between $\mathbb{E}\left\{F\left(\bar{\boldsymbol{w}}^{(T)}\right)\right\}$ and $F(\boldsymbol{w}^*)$, where $\bar{\boldsymbol{w}}^{(T)}$ is the final global parameters of FL system containing M potential unreliable clients and $\boldsymbol{w}^*$ is the optimal model parameters that minimizes $F(\boldsymbol{w})$.

Theorem 1 *For some $\varepsilon > 0$ and $\Theta > 0$, when the clients in the FL system behave unreliable with probability p_{U}, the convergence upper bound with a fixed total number of iterations T is given by*

$$\mathbb{E}\left\{F\left(\bar{\boldsymbol{w}}^{(T)}\right)\right\} - F(\boldsymbol{w}^*)$$

$$\leq \frac{1}{T\left(\omega\eta(1 - \frac{\beta\eta}{2}) - \frac{\rho\left(\phi(\tau) + \frac{p_{\mathrm{U}}}{M}\left[(1-\alpha)M\Theta + \frac{2\sqrt{M}\sigma}{\pi}\right]\right)}{\tau\varepsilon^2}\right)}, \tag{6}$$

where $\phi(\tau) = \frac{\delta}{\beta}\left((\eta\beta + 1)^\tau - 1\right) - \eta\delta\tau$, $\varphi = \omega(1 - \frac{\beta\eta}{2})$, and $\omega \triangleq \min_k \frac{1}{\|\boldsymbol{w}^{(k\tau)} - \boldsymbol{w}^*\|^2}$.

Proof: See Appendix A. ∎

The upper bound given by **Theorem 1** demonstrates the convergence result of the FL system with unreliable clients. A lower bound means that the value of the system loss function converges closer to the optimal one.

B. Discussions on the Convergence Bound

In this subsection, we will provide several key observations on the convergence bound.

Proposition 1 *If there is no unreliable client, the convergence bound of FL increases, which also means a **worse** system performance, as the local epochs τ increases. Since other parameters are basically fixed, the influence of τ on this theoretical value is the most noteworthy.*

Proof: When $p_U = 0$, the convergence upper bound can be expressed as $1/T\left(\omega\eta(1 - \frac{\beta\eta}{2}) - \frac{\rho(\phi(\tau))}{\tau\varepsilon^2}\right)$. It is evident that $\frac{\rho(\phi(\tau))}{\tau\varepsilon^2}$ increases with an larger τ and leads to a larger bound. ∎

Proposition 2 *When the probability of unreliable behaviors p_U is larger, the convergence performance becomes worse. However, when this percentage is fixed, the performance of the system will **improve** with the number of total clients M.*

Proof: Considering the analytical part in the convergence bound related to M, we find the value $\frac{p_U}{M}\left[(1-\alpha)M\Theta + \frac{2\sqrt{M}\sigma}{\pi}\right]$ decreases with the increase of M when p_U is fixed, thus the convergence bound becomes smaller. ∎

We note that when the total number of iterations T is constant, the local training iterations τ should be as small as possible if there is no unreliable client. This is because that when $\tau = 1$, the FL system based on distributed gradient descent is equivalent to a centralized training model [24]. However, in an unreliable circumstance, there exists an optimal value of $\tau \in [1, T]$ (T is an integer multiple of τ) to have the optimal convergence performance. Therefore, we can make the following proposition.

Proposition 3 *Under the unreliable behaviors of clients with a fixed T, the convergence upper-bound is a convex function of the number of local epochs τ, if we treat τ as a continuous variable.*

Proof: See Appendix B. ∎

From **Proposition 3**, we can see that there exists an optimal τ which can minimize the value of the loss function to obtain a satisfied learning performance.

V. DEFENSIVE MECHANISM DESIGN

In this section, we will use a crafted DNN to detect the existence of unreliable clients. Current defensive mechanisms, such as Krum [26] and Secprobe [27], usually need an online testing dataset to adjust the aggregation weight. The testing dataset is either from clients, which may pose privacy issue, or using a public one that may affect the accuracy. Thus, in this work we consider training an offline detector to recognise the abnormal clients. A basic binary anomaly detection technique using DNN operates in two steps. First, the DNN is trained on the normal training data to learn all normal labels. Second, each test instance is provided as an input to the DNN. If the DNN accepts the test input, it is labeled as normal and if the network rejects a test input, it is an anomaly. Therefore, we propose the DeepSA algorithm based on a crafted DNN in one-class setting. The main implemental process is operated in the server with a new functional module. To complete this module, the detector is pre-trained before FL with several normal parameter inputs, and these parameters can be obtained from clients or a public dataset. Once the pre-training process ends, this module can be used for anomaly detection.

In FL, the set of local models received by the server at the k-th communication round can be expressed as

$$\boldsymbol{o}^{(k)} \triangleq \left\{\boldsymbol{w}_1^{(k\tau)}, \boldsymbol{w}_2^{(k\tau)}, \ldots, \boldsymbol{w}_i^{(k\tau)}, \ldots, \boldsymbol{w}_U^{(k\tau)}\right\}. \quad (7)$$

We use a DNN based anomaly detector, denoted by $\mathfrak{D}$, which can be viewed as a classifier to assign a label (normal or abnormal). Typically, the outputs produced by this detector, are one of the following two types: 1) *Scores*: Scoring techniques assign an detecting score to each instance, which is utilized to analyze the possibility of unreliable clients; 2) *Labels*: Techniques in this category assign a label (benign or malicious) to each test instance. In our DNN based detector, we define the detecting result of a test instance (or observation) by $\boldsymbol{z}^{(k)} = [z_1^{(k)}, z_2^{(k)}, \ldots, z_U^{(k)}] \in \{0, 1\}^U$. If $z_i = 0$, it represents that $\boldsymbol{w}_i^{(k\tau)}$ is the unreliable model. Therefore, the detecting process can be given by

$$\boldsymbol{z}^{(k)} = \mathfrak{D}\left(\boldsymbol{o}^{(k)}, \boldsymbol{o}^{(k-1)}, \ldots, \boldsymbol{o}^{(1)}\right), \quad (8)$$

where $\mathfrak{D}$ is the detector and we assume that whole sets of local models (from 1-st to k-th communication round) can be utilized as the input of this detector.

In detail, the server will receive the observations $\boldsymbol{o}^{(k)}$ at the k-th communication round, and try to identify these abnormal models in them. For a normal client, there should be a certain level of correlation among its uploaded parameters in consecutive communication rounds. However, manipulation of parameters by anomaly clients may break this correlation, hence the previous observations can also assist detecting abnormal models. Therefore, in order to enhance this difference, we use an input reshaping approach to ensure the shift-invariance property for our detector, which can be expressed as

$$\boldsymbol{O}^d = \begin{bmatrix} \boldsymbol{w}_1^{(k\tau)} & \boldsymbol{w}_1^{((k-1)\tau)} & \ldots & \boldsymbol{w}_1^{((k-d)\tau)} \\ \boldsymbol{w}_2^{(k\tau)} & \boldsymbol{w}_2^{((k-1)\tau)} & \ldots & \boldsymbol{w}_2^{((k-d)\tau)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{w}_{U-1}^{(k\tau)} & \boldsymbol{w}_{U-1}^{((k-1)\tau)} & \ldots & \boldsymbol{w}_{U-1}^{((k-d)\tau)} \\ \boldsymbol{w}_U^{(k\tau)} & \boldsymbol{w}_U^{((k-1)\tau)} & \ldots & \boldsymbol{w}_U^{((k-d)\tau)} \end{bmatrix}, \quad (9)$$

where d is the depth of the observation. The input $\boldsymbol{O}^d$ is shaped as a multi-dimensional vector ($U \times 1 \times d \times s_w$), where s_w is the size of the standard uploaded model. With this input design, we will introduce the construction of our DNN based detector.
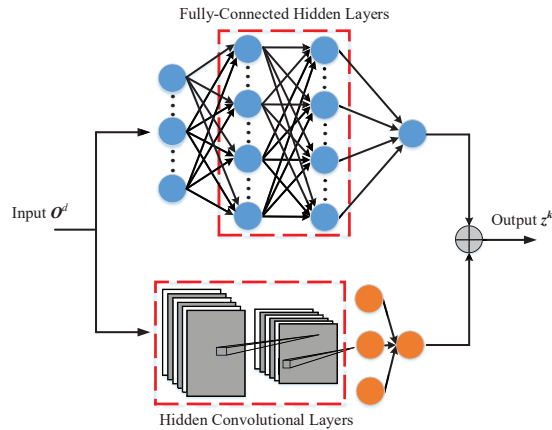

Fig. 2.   The architecture of the DNN based detector.

As shown in Fig. 2, the DNN detector consists of two parallel pipelines and achieves an output with the XOR operator. The input is $\boldsymbol{O}^k$ and the output is the symbol $\boldsymbol{z}^k$. Using the fully-connected layers and sufficient training, we can identify the intentional behaviors, such as the scaling operation on the parameters. In addition, for the randomized parameters, the convolutional layer can be more useful since the correlation between the noised and normal parameters is expected to be low. In the convolutional layer, we use zero-padding with stride size B, and set the filter size to $B \times l$, where l is the depth of the filter. This setting is based on the observation that each sub-vector is strongly correlated with $2B$ neighbouring sub-vectors due to the structure of uploaded models.

Algorithm 1 Secure Aggregation on the Server Side

Input: The observation of all uploaded models with d depth $\boldsymbol{O}^d$, the well trained DNN based detector $\mathfrak{D}$

Output: The global model $\boldsymbol{w}^{(k\tau)}$

1: Wait for clients to upload their weights until there are M clients' models $\boldsymbol{o}^k = \left\{\boldsymbol{w}_1^{(k\tau)}, \boldsymbol{w}_2^{(k\tau)}, \ldots, \boldsymbol{w}_i^{(k\tau)}, \ldots, \boldsymbol{w}_M^{(k\tau)}\right\}$

2: Update the input $\boldsymbol{O}^d$ with the fresh uploaded models $\boldsymbol{o}^k$

3: Obtain the detecting results with the DNN based detector $\mathfrak{D}$ as $\boldsymbol{z}^k = \mathfrak{D}\left(\boldsymbol{O}^d\right)$

4: Average the benign models and obtain $\boldsymbol{w}^{(k\tau)} = \sum_{i=1}^{M} z_i p_i \boldsymbol{w}_i^{(k\tau)}$

5: Send the new averaged model $\boldsymbol{w}^{(k\tau)}$ to all clients

6: **return** $\boldsymbol{w}^{(k\tau)}$

After introducing this crafted DNN based detector, we present our proposed defence algorithm as shown in **Algorithm 1**. As discussed above, the existence of abnormal clients indicates that the parameters uploaded by them may be disruptive, and it may reduce the accuracy of the global model. To mitigate their effect on the model accuracy, we remove the malicious models in this communication rounds using our DNN based detector.

Algorithm 1 gives the pseudocode of secure aggregation on the server side. The server first waits for the local model from each client. When all the clients finish uploading their models to the server, these models are utilized to update the input $\boldsymbol{O}^d$ with the fresh uploaded models. Then, the server can obtain the detecting results with the DNN based detector and average the benign models. For the whole system, reducing the number of clients is equivalent to reducing the amount of training data, which will reduce the generalization of the global model. However, compared with the damage brought by unreliable clients, these losses are acceptable. In addition, considering that even a reliable client may upload a model with poor quality, the decision result of DNN detector will only take effect in the current round of communication.

VI. EXPERIMENTAL RESULTS

In this section, we first evaluate the performance of the analytical results with the unreliable clients, and verify the effectiveness with the experimental results. Then, we demonstrate the effectiveness of the proposed defensive mechanism by comparing with other algorithms[2].

A. Experimental Settings

1) Dataset: In our experimental results, we use four benchmark datasets for different tasks:

- MNIST and Fashion-MNIST dataset, which both have $70K$ digit images of size 28×28, are split into $60K$ training and $10K$ test samples;
- CIFAR-10 dataset, which consists of $60K$ color images in 10 object classes such as deer, airplane, and dog with 6000 images included per class, are split into $50K$ training and $10K$ test samples;
- Adult dataset, which has around $32K$ tabular samples and each sample has 14 attributes, is split into $20K$ training and $12K$ test samples.

We consider the data distribution as independent identically distributed (i.i.d), i.e. clients in the FL system possess a same amount of data from training sets randomly and independently.

2) Parameter Settings: We use the multi-layer perceptron (MLP) as the training model to construct the FL system, and each client locally computes stochastic gradient descend (SGD) updates on each dataset, and then aggregate updates to train a globally shared classifier. We conduct three cases for the unreliable client to verify the analytical results as follows:

- Case I: $\alpha = -1$ and $\sigma = 0.1$, which an unreliable client uploads a completely inverse parameter with small noise;
- Case II: $\alpha = 0.8$, and $\sigma = 0.5$, which a large noise is added;
- Case III: $\alpha = 0.5$, and $\sigma = 0.3$, which the parameter is scaling half with a medium noise.

In addition, we set the total number of clients $M = 50$ and the total learning iterations $T(k\tau) = 300$. We run each experiment for 20 times and record the average results. If

---

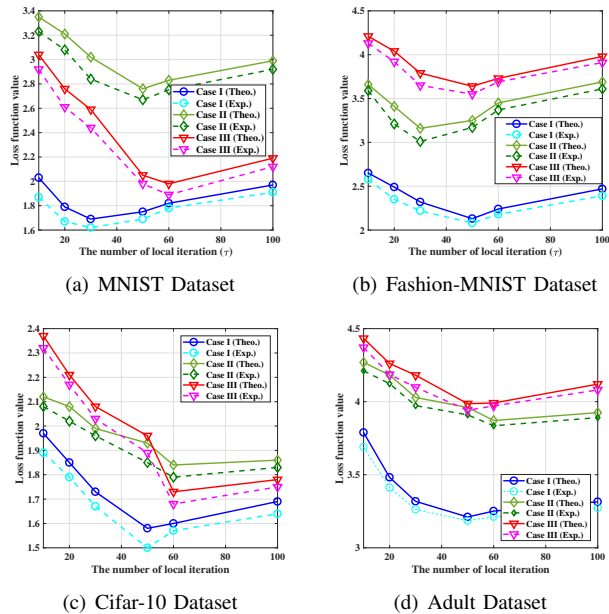[2]Related codes can be found in the following link: https://github.com/JJisbug/UnreliableClientsinFL.

Fig. 3. Comparison of the loss function value between the theoretical and experimental results.

Fig. 4. The classification accuracy of local iterations with a certain probability of unreliable behavior.

a client uploads unreliable parameters in all communication rounds, then this scenario can be treated as a special case in which we assume there are certain percentages of unreliable clients, and other clients will upload reliable parameters during the whole training process.

3) Comparing defensive mechanisms: To show the effectiveness of the proposed defensive mechanisms, we provide the following defensive mechanisms:

- Krum [26]: the aggregated parameters are chosen according to the minimum geometric gradient rule.
- Secprobe [27]: the aggregated parameters are chosen according to the testing accuracy.
- Pearson [28]: the aggregated weights are adjusted by the Pearson correlation.

B. Theoretical Results

In Fig. 3, we show the experimental results of loss function value as a function of τ with $p_U = 0.1$ under the unreliable environment. In order to be close to reality (the local training epoch (τ) and communication rounds (k) of clients are not too small) the range of τ is set to $[10, 100]$. We can observe that the theoretical bounds are convex functions and close to the real results for the three cases and four datasets, which are consistent with **Theorem 1** and **Proposition 3**. The reason behind this phenomenon is that a large local epochs τ will decrease the times of uploading unreliable parameters, while a small τ incurs much unreliability in the parameters uploaded by all clients. In addition, with a smaller value of added noise, the learning performance will get fewer negative influences.

C. Experimental Results with Unreliable Behavior

In this subsection, we show the classification accuracy based on the FL system with different probabilities of unreliable
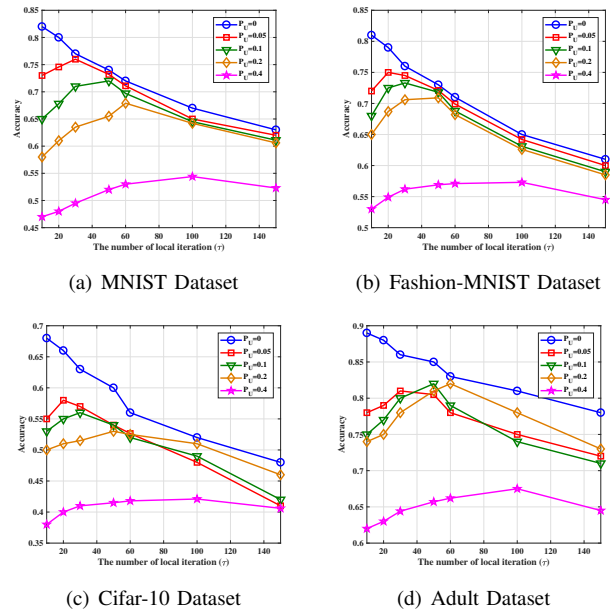
clients in Fig. 4. We take **Case I** as the abnormal client for MNIST and Fashion-MNIST dataset, **Case II** for Cifar-10 dataset and **Case III** for Adult dataset, respectively. In order to show different conditions, we also set the probability of unreliable clients p_U to 0.05, 0.1, 0.2 and 0.4, respectively. From these figures we find that when there is no abnormal client ($p_U = 0$), the system performance decreases with the increase of the local iteration τ, which is consistent with **Proposition 1**. However, when the uploading environment is unreliable, we can note that there exits an optimal number of the local iterations τ in terms of system performance, which is in line with **Proposition 3**. We can also note that the optimal number of local training iterations increases with the probability of abnormal clients. The intuition is that more communication rounds will produce a larger damage to the FL system, but more communication rounds also bring a better performance for a normal FL system. In addition, we find that, with an increasing probability of unreliable probability, the system performance shows a descending trend, and a system with relatively high probability, i.e., $p_U > 50\%$, may fall to converge.

In Fig. 5, we show the loss function value under different numbers of total clients that we set M to 50, 100, 150, 200 and 250. In Fig. 5(a) we use **Case I**, in Fig. 5(b)-5(c) we use **Case II**, and in Fig. 5(d) we use **Case III**, respectively. We can note that the system has a better performance with a smaller unreliable probability (p_U), and the loss function value in both figures keeps decreasing with the number of total clients, which is consistent with **Proposition 2**.

D. Performance of the Proposed DeepSA Algorithm

In this subsection, we conduct experiments on our proposed DeepSA based federated training against various percentages of unreliable clients. We use ReLU as the activation functions
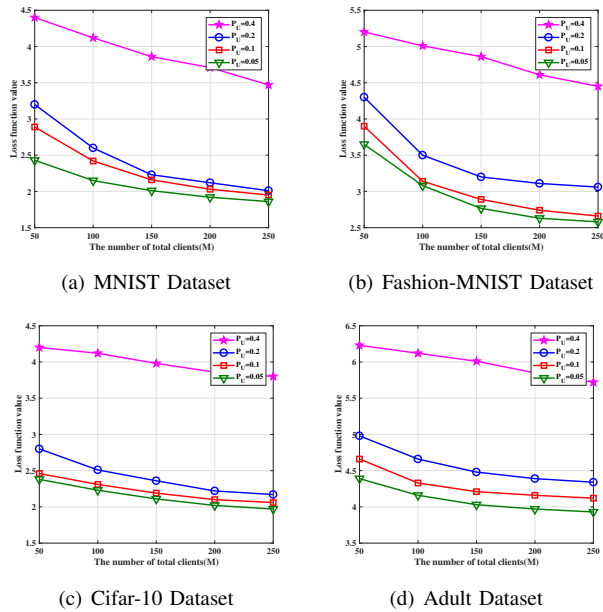
Fig. 5.   The loss function value with different number of total clients.

Fig. 6.   The successful detecting rate with a trained DNN detector under a federated model against different probabilities of unreliable clients p_{U}.

for the hidden layers: $y = \mathrm{ReLU}(x) = \max(x, 0)$, where $x \in \mathbb{R}$ is the input, and y is the output of the activation function. To map the output to the interval between $(0, 1)$, we choose the sigmoid function as the activation function for the output layer: $y = \mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}}$. In Fig. 6 we show the detecting results with a stable scalar $\alpha = 0.8$, and various noises. From this figure we can observe that with a larger standard deviation σ, the trained DNN based detector will perform better in identifying these unreliable clients. This is because with a larger standard deviation, a more obvious difference of parameters from neighbouring communication rounds will be recognized by the trained detector. Moreover, it can be noted that when $\sigma > 0.22$, this detector will have an excellent performance, which can guarantee no errors, i.e., the detecting rate is 1. We can also find that if p_{U} is larger, the successful detecting rate will decrease which means that it is more difficult for detectors to identify.

Fig. 7 show the comparison results between the proposed DeepSA algorithm and others, in which we set $p_{\mathrm{U}} = 0.2$ and unreliable behaviours consider Case I, II and III. In addition, in Table II and III we consider a more practical scenario in which clients may behave unreliably with different probabilities. In details, we assume there are 100 clients which are divided into 4 equal-size groups. The probabilities of unreliable behaviours for the 4 groups are set to 0.1, 0.2, 0.3 and 0.4, respectively. It is obvious that with our proposed DeepSA algorithm, the federated training process perform better in most cases. The reason is that the similarity based detection algorithms (Secprobe and Pearson) can only handle the noise perturbation behavior, and Krum loses its performance due to the limited number of participants, while the proposed algorithm has a high detecting rate which enhances the learning performance.

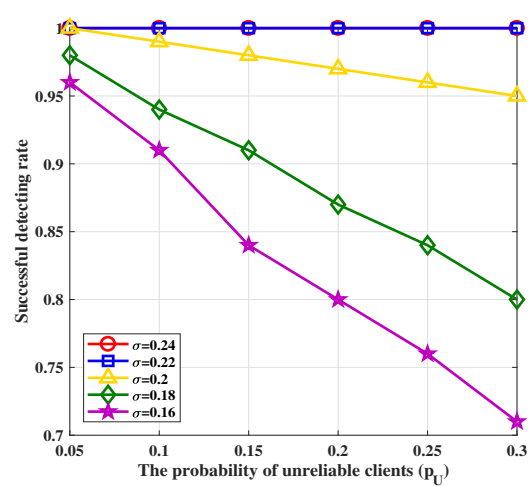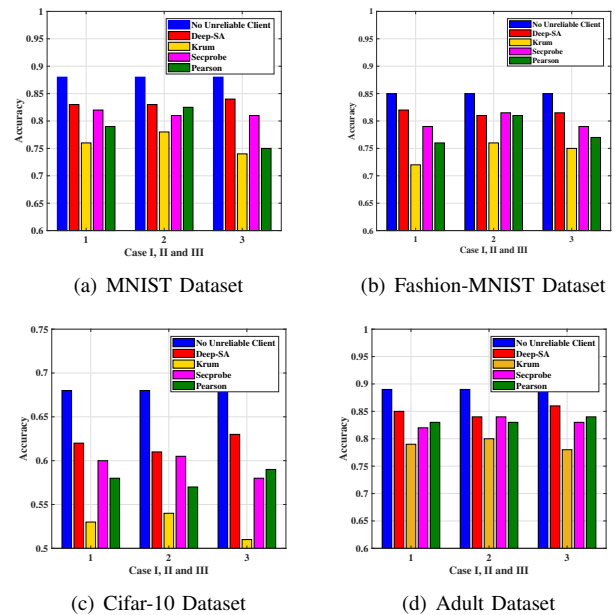In addition, the proposed defensive mechanism is applied

Fig. 7.   The classification accuracy comparison between the proposed Deep-SA algorithm and others.

to four real-world datasets, i.e., Sports[3], UAV Detection[4],

[3]https://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities.
[4]https://archive.ics.uci.edu/ml/datasets/Unmanned+Aerial+Vehicle+%28UAV%29+Intrusion+Detection.

TABLE II
THE CLASSIFICATION ACCURACY COMPARISON IN MNIST/FASHION MNIST DATASET WITH DIFFERENT UNRELIABLE PROBABILITIES.

	All reliable	Deep-SA	Krum	Secprobe	Pearson
Case I	0.88/0.85	**0.81/0.78**	0.71/0.69	0.78/0.75	0.74/0.71
Case II	0.88/0.85	**0.8/0.765**	0.72/0.69	0.77/0.74	0.76/0.72
Case III	0.88/0.85	**0.81/0.77**	0.73/0.7	0.78/0.75	0.74/0.7

TABLE III
THE CLASSIFICATION ACCURACY COMPARISON IN CIFAR-10/ADULT
DATASET WITH DIFFERENT UNRELIABLE PROBABILITIES.

	All reliable	Deep-SA	Krum	Secprobe	Pearson
Case I	0.68/0.89	**0.58/0.82**	0.51/0.77	0.57/0.8	0.54/0.78
Case II	0.68/0.89	**0.59/0.83**	0.52/0.79	0.57/0.81	0.56/0.79
Case III	0.68/0.89	**0.6/0.83**	0.52/0.78	0.58/0.81	0.54/0.78

(a) Sports Dataset

(b) UAV Detection Dataset

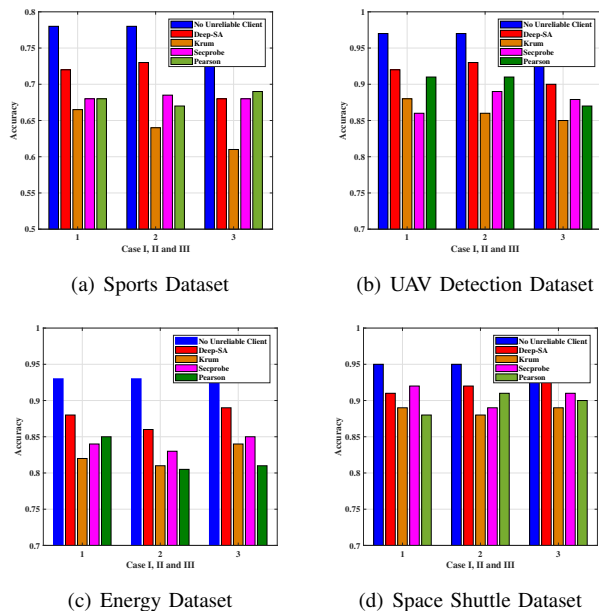(c) Energy Dataset

(d) Space Shuttle Dataset

Fig. 8. The classification accuracy comparison between the proposed Deep-SA algorithm and others.

Energy[5] and Space Shuttle[6], which have been collected from real-life sensors [29], [30], and the descriptions of these datasets are listed as follows:

- Sports: This dataset comprises motion sensor data of 19 daily and sports activities each performed by 8 subjects in their own style for 5 minutes, and we evaluate the performance by the accuracy of a 19-class classifier.
- UAV Detection: This dataset consists of 55 attributes in which each data row represents an encrypted WiFi traffic record. The output shows the current traffic is from a UAV or not.
- Energy: This dataset consists of assessing the heating load and cooling load requirements of buildings (that is, energy efficiency) as a function of building parameters with 8 attributes, and aims to predict each of two responses.
- Space Shuttle: This shuttle dataset contains 9 attributes and 58000 numerical instances with an 80% default accuracy.

In Fig. 8, we verify the proposed defensive algorithm under Case I, II and III of unreliable clients. As can be found in this figure, although some algorithms may have a slightly better or equal performance than the proposed algorithm, Deep-SA outperforms other algorithms in most cases. For example, in

[5]https://archive.ics.uci.edu/ml/datasets/Energy+efficiency.
[6]https://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle).

the Sports dataset, although DeepSA has not achieved the top performance in Case III (68% v.s. 69%), it has a performance gain in other cases (4% in Case I and 4.5% in Case II, respectively).

VII. RELATED WORKS

In this section, we investigate different adversarial models in FL and defensive mechanisms, which are active areas of research.

A. Adversarial Models in Federated Learning

The security of machine learning has attracted heated attentions recently [31], [32]. Although the data is not explicitly exposed in the original format in distributed learning frameworks, e.g., FL [10], different types of adversarial models against distributed machine learning algorithms have been designed and analyzed including poisoning attacks (e.g., [21], [33]) and privacy attacks (e.g., [34]–[36]). For example, poisoning attackers can control part of clients and manipulate the outputs sent to the server, which can mislead the global model deviate to the designed direction [21]. The authors in [14] proposed a novel attacking method that a malicious opponent may interfere with the learning process by applying limited changes to the uploaded models. In addition, the authors in [33] explored the adversarial of model poisoning attacks on FL, which supported by a single, non-colluding malicious client where the adversarial objective is to make the global model misclassify a set of chosen inputs with high confidence.

B. Defensive Mechanisms

With the development of adversarial models in FL, how to design an effective defensive mechanism to defeat these malicious clients has became crucial. For detecting poisoned updates in the collaborative learning [16], the results of client-side cross-validation were applied for adjusting the weights of the updates when performing aggregation, where each update is evaluated over other clients' local data. Similar approach based on Pearson similarity is proposed in [28]. In addition, the authors [27], [37] in considered the existence of unreliable clients in Fl and used the auxiliary validation data to compute a utility score for each participant, thus reducing the negative impact of these unreliable participants. The work in [26] proposed a novel poisoning defensive method in Fl. In details, for each client, the server will calculate the sum of the Euclidean distances to the models of other clients, and select the one with the minimum sum. However, the mentioned defensive algorithms all need an online detection process while the access to the auxiliary dataset may leak privacy.

VIII. CONCLUSION

In this work, we have introduced a new threat model of adversary clients for FL systems. By deriving convergence bound on the loss function of the trained FL model, we have seen that there exists an optimal number of local training iterations to achieve the best performance with a fixed total amount of computing resources. Furthermore, we have designed a

novel defensive algorithm using the DNN detection technique, termed DeepSA, which can automatically detect unreliable models and remove them from the aggregation process. Extensive experimental results have validated our analysis and the effectiveness of the proposed DeepSA algorithm.

APPENDIX A
PROOF OF THEOREM 1

When we consider the unreliable behavior and Eq. (5), we have

$$\bar{\boldsymbol{w}}^{(k\tau)} = \frac{1}{M} \sum_{i \in M} \left[(1 - p_{\text{U}})\boldsymbol{w}_i + p_{\text{U}}\hat{\boldsymbol{w}}_i \right]. \tag{10}$$

Then, according to **Assumption 1**, the difference between $F(\bar{\boldsymbol{w}}(k\tau))$ and $F(\boldsymbol{w}(k\tau))$ can be expressed as

$$
\begin{aligned}
F(\bar{\boldsymbol{w}}^{(k\tau)}) - F(\boldsymbol{w}^{(k\tau)}) &\leq \rho \left\| \bar{\boldsymbol{w}}^{(k\tau)} - \boldsymbol{w}^{(k\tau)} \right\| \\
&= \frac{\rho}{M} \left\| \sum_{i \in M} p_{\text{U}} (\hat{\boldsymbol{w}}_i - \boldsymbol{w}_i) \right\| \\
&= \frac{\rho}{M} \left\| \sum_{i \in M} p_{\text{U}} [(\alpha - 1)\boldsymbol{w}_i + \boldsymbol{n}_i] \right\| \\
&\leq \frac{\rho p_{\text{U}}}{M} \left[\mathbb{E} \left\| \sum_{i \in M} (\alpha - 1)\boldsymbol{w}_i \right\| + \mathbb{E} \left\| \sum_{i \in M} \boldsymbol{n}_i \right\| \right] \\
&\leq \frac{\rho p_{\text{U}}}{M} \left[(1 - \alpha)M\Theta + \frac{2\sqrt{M}\sigma}{\pi} \right],
\end{aligned}
\tag{11}
$$

where $\|\|$ denotes the L_2 norm function, Θ is a upper bound of all model parameters, respectively. For simplicity, we omit the superscript $(k\tau)$ of $\boldsymbol{w}_i$ and $\boldsymbol{n}_i$.

We define $\boldsymbol{v}_{[k]}^{(t)}$ as an auxiliary parameter vector, which follows an centralized gradient decent.

$$\boldsymbol{v}_{[k]}^{(t)} = \boldsymbol{v}_{[k]}^{(t-1)} - \eta \nabla F(\boldsymbol{v}_{[k]}^{(t-1)}). \tag{12}$$

The notation $[k]$ defines the interval $[(k-1)\tau, k\tau]$ for $k = 1, 2, \ldots, K$, and the auxiliary parameter vector $\boldsymbol{v}_{[k]}^{(t)}$ only works in this interval. We denote by $\boldsymbol{v}_{[k]}^{(k\tau)}$ and $\boldsymbol{v}_{[k+1]}^{(k\tau)}$ the two different auxiliary vectors before and after the kth aggregation respectively, which can be seen in Fig. 9. At the beginning of the interval $[k]$, $\boldsymbol{v}_{[k]}^{(t)}$ will not inherit the last result of the previous interval but is equivalent to the global parameter after kth aggregation. i.e., $\boldsymbol{v}_{[k]}^{(k\tau)} \triangleq \boldsymbol{w}^{(k\tau)}$.

Using the upper bound in [24], we know that

$$F(\boldsymbol{w}^{(k\tau)}) - F(\boldsymbol{v}_{[k]}^{(k\tau)}) \leq \rho \phi(\tau), \tag{13}$$

where k is the index of the aggregation, and $\phi(\tau) = \frac{\delta}{\beta} \left((\eta\beta + 1)^\tau - 1 \right) - \eta\delta\tau$.

Combing Eq. (11) with Eq. (13) we can obtain

$$
\begin{aligned}
F(\bar{\boldsymbol{w}}^{(k\tau)}) - F(\boldsymbol{v}_{[k]}^{(k\tau)}) &\leq \rho \phi(\tau) + \frac{\rho p_{\text{U}}}{M} \left[(1 - \alpha)M\Theta + \frac{2\sqrt{M}\sigma}{\pi} \right] \\
&= \rho \left\{ \phi(\tau) + \frac{p_{\text{U}}}{M} \left[(1 - \alpha)M\Theta + \frac{2\sqrt{M}\sigma}{\pi} \right] \right\} = \rho\Delta,
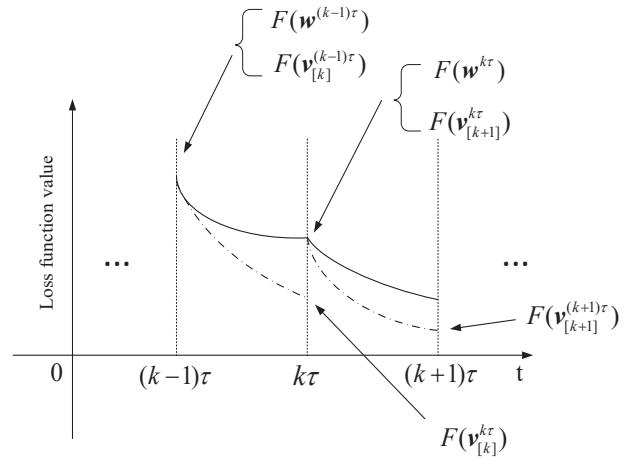\end{aligned}
\tag{14}
$$

Fig. 9. The auxiliary parameter vector in FL.

where $\phi(\tau) + \frac{p_{\text{U}}}{M} \left[(1 - \alpha)M\Theta + \frac{2\sqrt{M}\sigma}{\pi} \right]$ is denoted by Δ. Then, we define $\theta_{[k]}^{(t)} = F\left(\boldsymbol{v}_{[k]}^{(t)}\right) - F(\boldsymbol{w}^*)$ for an interval $[k]$ where k is fixed, t is defined between $(k-1)\tau \leq t \leq k\tau$. According to **Assumption 1** and [24], we have

$$\theta_{[k]}^{(t)} > \varepsilon, \tag{15}$$

and

$$
\begin{aligned}
\frac{1}{\theta_{[k]}^{(T)}} - \frac{1}{\theta_{[1]}^{(0)}} \geq {}& \sum_{k=1}^{K-1} \left(\frac{1}{\theta_{[k+1]}^{(k\tau)}} - \frac{1}{\theta_{[k]}^{(k\tau)}} \right) \\
&+ T\omega\eta \left(1 - \frac{\beta\eta}{2} \right).
\end{aligned}
\tag{16}
$$

According to the definition of $\boldsymbol{v}_{[k+1]}^{(k\tau)}$, we have

$$\boldsymbol{v}_{[k+1]}^{(k\tau)} = \bar{\boldsymbol{w}}^{(k\tau)}. \tag{17}$$

Then, we have

$$
\begin{aligned}
\frac{1}{\theta_{[k+1]}^{(k\tau)}} - \frac{1}{\theta_{[k]}^{(k\tau)}} &= \frac{\theta_{[k]}^{(k\tau)} - \theta_{[k+1]}^{(k\tau)}}{\theta_{[k]}^{(k\tau)} \theta_{[k+1]}^{(k\tau)}} = \frac{F\left(\boldsymbol{v}_{[k]}^{(k\tau)}\right) - F\left(\boldsymbol{v}_{[k+1]}^{(k\tau)}\right)}{\theta_{[k]}^{(k\tau)} \theta_{[k+1]}^{(k\tau)}} \\
&= \frac{F\left(\boldsymbol{v}_{[k]}^{(k\tau)}\right) - F\left(\bar{\boldsymbol{w}}^{(k\tau)}\right)}{\theta_{[k]}^{(k\tau)} \theta_{[k+1]}^{(k\tau)}} \geq -\frac{\rho\Delta}{\varepsilon^2}.
\end{aligned}
\tag{18}
$$

Combining Eq. (16) with Eq. (18), we have

$$\frac{1}{\theta_{[k]}^{(T)}} - \frac{1}{\theta_{[1]}^{(0)}} \geq -\frac{\rho(K-1)\Delta}{\varepsilon^2} + T\omega\eta \left(1 - \frac{\beta\eta}{2} \right). \tag{19}$$

When $k = K$, according to the **Assumption 1**, we can obtain

$$F(\hat{\boldsymbol{w}}^{(T)}) - F(\boldsymbol{w}^*) \geq \varepsilon. \tag{20}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3079472, IEEE Internet of Things Journal

10

Therefore, Eq. (20) can be expressed as

$$
\frac{1}{F\left(\bar{\boldsymbol{w}}^{(T)}\right) - F\left(\boldsymbol{w}^*\right)} - \frac{1}{\theta_{[K]}^{(T)}}
$$

$$
= \frac{\theta_{[K]}^{(T)} + F\left(\boldsymbol{w}^*\right) - F\left(\bar{\boldsymbol{w}}^{(T)}\right)}{\left(F(\bar{\boldsymbol{w}}^{(T)}) - F(\boldsymbol{w}^*)\right)\theta_{[K]}^{(T)}} \tag{21}
$$

$$
= \frac{F(\boldsymbol{v}_{[k]}^{(T)}) - F(\bar{\boldsymbol{w}}^{(T)})}{\left(F(\bar{\boldsymbol{w}}^{(T)}) - F(\boldsymbol{w}^*)\right)\theta_{[K]}^{(T)}} \geq \frac{-\rho\Delta}{\varepsilon^2}.
$$

Summing up equation (19) and (21), we have

$$
\frac{1}{F(\bar{\boldsymbol{w}}^{(T)}) - F(\boldsymbol{w}^*)} - \frac{1}{\theta_{[1]}^{(0)}}
$$

$$
= T\omega\eta\left(1 - \frac{\beta\eta}{2}\right) - \frac{\rho K\Delta}{\varepsilon^2} \tag{22}
$$

$$
\overset{(a)}{=} T\left(\omega\eta\left(1 - \frac{\beta\eta}{2}\right) - \frac{\rho\Delta}{\tau\varepsilon^2}\right),
$$

where step (a) is obtained by $T = K\tau$.

Note that $\theta_{[1]}^{(0)} > 0$, the above inequality can be simplified as

$$
\frac{1}{F(\widehat{\boldsymbol{w}}^{(T)}) - F(\boldsymbol{w}^*)} \geq T\left(\omega\eta(1 - \frac{\beta\eta}{2}) - \frac{\rho\Delta}{\tau\varepsilon^2}\right). \tag{23}
$$

According to the definition we know that $\boldsymbol{w}^*$ is the optimal model parameters minimizing the $F(\boldsymbol{w})$. Hence $F(\widehat{\boldsymbol{w}}_{[k]}^{(T)}) \geq F(\boldsymbol{w}^*)$ and this inequality can be true when the right-hand side of the inequality is less than 0. Note that when the upper bound of parameter Θ or the unreliable p_U or the additive noise power σ is big enough, $T\left(\omega\eta(1 - \frac{\beta\eta}{2}) - \frac{\rho\left(\phi(\tau) + \frac{p_U}{M}\left[(1-\alpha)M\Theta + \frac{2\sqrt{M}\sigma}{\pi}\right]\right)}{\tau\varepsilon^2}\right)$ will less than zero. In this case, although the inequality (23) is true or not true it will make no any sense. This can be interpreted as that the system will crash when the learning circumstance is unacceptable. Similarly, when local training epoch τ continues to increase without limitation, the right-hand side of this inequality will be smaller than zero. Therefore, for ease of analysis we assume there are certain limits on Θ, p_U, σ and τ. In other words we assume that $T\left(\omega\eta(1 - \frac{\beta\eta}{2}) - \frac{\rho\Delta}{\tau\varepsilon^2}\right) > 0$. Then taking the reciprocal of the above inequality yields

$$
F(\widehat{\boldsymbol{w}}^{(T)}) - F(\boldsymbol{w}^*)
$$

$$
\leq \frac{1}{T\left(\omega\eta(1 - \frac{\beta\eta}{2}) - \frac{\rho\left(\phi(\tau) + \frac{p_U}{M}\left[(1-\alpha)M\Theta + \frac{2\sqrt{M}\sigma}{\pi}\right]\right)}{\tau\varepsilon^2}\right)}. \tag{24}
$$

This completes the proof. □

APPENDIX B
PROOF OF PROPOSITION 3

First, we define $H(\tau)$ as

$$
H(\tau) \triangleq \frac{\phi(\tau) + \nabla}{\tau}, \tag{25}
$$

where $\phi(\tau) = \frac{\delta}{\beta}\left((\eta\beta + 1)^\tau - 1\right) - \eta\delta\tau$ and $\nabla = \frac{p_U}{M}\left[(1-\alpha)M\Theta + \frac{2\sqrt{M}\sigma}{\pi}\right] > 0$, respectively.

With a slight abuse of τ, we consider continuous values of $\tau > 1$, and then have

$$
H'(\tau) = -\frac{\nabla}{\tau^2} + \frac{\delta\left(\eta\beta + 1\right)^\tau(\ln(\eta\beta + 1)^\tau - 1) + \delta}{\beta\tau^2}, \tag{26}
$$

and

$$
H''(\tau)
$$

$$
= \frac{\delta}{\beta\tau^4}\left(\tau^3(\eta\beta + 1)^\tau \ln^2(\eta\beta + 1)\right.
$$

$$
\left. -2\tau(\eta\beta + 1)^\tau(\ln(\eta\beta + 1)^\tau - 1) - 2\tau\right) + \frac{2\nabla}{\tau^3}
$$

$$
= \frac{\delta}{\beta\tau^3}\left((\eta\beta + 1)^\tau((\ln(\eta\beta + 1)^\tau - 1)^2 + 1) - 2\right) + \frac{2\nabla}{\tau^3}. \tag{27}
$$

We then define $f(x)$ as

$$
f(x) \triangleq x\left((\ln x - 1)^2 + 1\right) - 2, \tag{28}
$$

then, the derivate of $f(x)$ can be expressed as

$$
f'(x) = (\ln x - 1)^2 + 2(\ln x - 1) + 1
$$

$$
= (\ln x - 1 + 1)^2 \tag{29}
$$

$$
= \ln^2 x \geq 0.
$$

Note that $(\eta\beta + 1)^\tau \geq 1$ since $\tau \geq 1, \eta\beta \geq 0$, we can know that

$$
f(\eta\beta + 1)^\tau \geq f(1) = 0. \tag{30}
$$

Combining (27), (28) and (30), we have

$$
H''(\tau) \geq 0. \tag{31}
$$

Therefore, under the unreliable behaviors of clients with a fixed T, the convergence upper-bound is a convex function of the number of local epochs τ. This completes the proof. □

REFERENCES

[1] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[2] J. Li, S. Chu, F. Shu, J. Wu, and D. N. K. Jayakody, "Contract-based small-cell caching for data disseminations in ultra-dense cellular networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 5, pp. 1042–1053, 2019.

[3] S. Shaham, M. Ding, B. Liu, S. Dang, Z. Lin, and J. Li, "Privacy preserving location data publishing: A machine learning approach," *IEEE Transactions on Knowledge and Data Engineering*, no. to appear, 2020.

[4] S. Shaham, M. Ding, B. Liu, S. Dang, Z. Lin, and J. Li, "Privacy preservation in location-based services: A novel metric and attack model," *IEEE Transactions on Mobile Computing*, to appear 2020.

[5] D. C. Nguyen, P. Cheng, M. Ding, D. Lopez-Perez, P. N. Pathirana, J. Li, A. Seneviratne, Y. Li, and H. V. Poor, "Enabling ai in future wireless networks: A data life cycle perspective," *IEEE Communications Surveys Tutorials*, vol. 23, no. 1, pp. 553–595, 2021.

[6] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, and A. Dubey, "No Peek: A survey of private distributed deep learning," *arXiv preprint arXiv:1812.03288*, 2018.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 54, Fort Lauderdale, USA, Apr. 2017, pp. 1273–1282.

[8] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[9] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE access*, vol. 6, pp. 6900–6919, 2017.

[10] C. Ma, J. Li, M. Ding, H. H. Yang, F. Shu, T. Q. S. Quek, and H. V. Poor, "On safeguarding privacy and security in the framework of federated learning," *IEEE Network*, pp. 1–7, 2020.

[11] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[12] Y. Peng, A. Tan, J. Wu, and Y. Bi, "Hierarchical edge computing: A novel multi-source multi-dimensional data anomaly detection scheme for industrial internet of things," *IEEE Access*, vol. 7, pp. 111 257–111 270, 2019.

[13] H. Peng, S. Si, M. K. Awad, N. Zhang, H. Zhao, and X. S. Shen, "Toward energy-efficient and robust large-scale wsns: A scale-free network approach," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 4035–4047, 2016.

[14] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 8632–8642.

[15] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 108, Palermo, Sicily, Italy, Aug. 2020, pp. 2938–2948.

[16] L. Zhao, S. Hu, Q. Wang, J. Jiang, S. Chao, X. Luo, and P. Hu, "Shielding collaborative learning: Mitigating poisoning attacks through client-side detection," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020.

[17] S. Fu, C. Xie, B. Li, and Q. Chen, "Attack-resistant federated learning with residual-based reweighting," *CoRR*, vol. abs/1912.11464, 2019. [Online]. Available: http://arxiv.org/abs/1912.11464

[18] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1486–1500, 2020.

[19] Y. Zhao, J. Chen, J. Zhang, D. Wu, J. Teng, and S. Yu, "PDGAN: A novel poisoning defense method in federated learning using generative adversarial network," in *Proc. Algorithms and Architectures for Parallel Processing*, Cham, 2020, pp. 595–609.

[20] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," *arXiv*, Sep. 2019. [Online]. Available: https://arxiv.org/abs/1909.05125v1

[21] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proc. 29th USENIX Security Symposium (USENIX Security 20)*, Aug. 2020.

[22] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[23] F. Zhou and G. Cong, "On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization," in *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, Stockholm, Sweden, jul. 2018, pp. 3219–3227.

[24] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[25] S. U. Stich, "Local SGD converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, 2018.

[26] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems, 2017*, pp. 119–129.

[27] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1486–1500, 2020.

[28] E. D. B. Solis, A. M. Neto, and B. N. Huallpa, "Real-time collision risk estimation based on pearson's correlation coefficient: Comparative analysis with real distance from the velodyne 3d laser scanner," in *Proc. 2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, 2016, pp. 234–238.

[29] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[30] T. Luo and S. G. Nagarajan, "Distributed anomaly detection using autoencoder neural networks in WSN for IoT," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[31] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *29th International Conference on Machine Learning*. ArXiv e-prints, 2012, pp. 1807–1814.

[32] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 4, pp. 984–996, 2013.

[33] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[34] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 603–618.

[35] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE INFOCOM 2019 Conference on Computer Communications*. IEEE, 2019, pp. 2512–2520.

[36] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 691–706.

[37] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1486–1500, 2019.