Byzantine-Robust Decentralized Federated Learning

Minghong Fang*
University of Louisville

Zifan Zhang
North Carolina State University

Hairi
University of Wisconsin-Whitewater

Prashant Khanduri
Wayne State University

Jia Liu
The Ohio State University

Songtao Lu
IBM Thomas J. Watson Research Center

Yuchen Liu
North Carolina State University

Neil Gong
Duke University

ABSTRACT

Federated learning (FL) enables multiple clients to collaboratively train machine learning models without revealing their private training data. In conventional FL, the system follows the server-assisted architecture (server-assisted FL), where the training process is coordinated by a central server. However, the server-assisted FL framework suffers from poor scalability due to a communication bottleneck at the server, and trust dependency issues. To address challenges, decentralized federated learning (DFL) architecture has been proposed to allow clients to train models collaboratively in a serverless and peer-to-peer manner. However, due to its fully decentralized nature, DFL is highly vulnerable to poisoning attacks, where malicious clients could manipulate the system by sending carefully-crafted local models to their neighboring clients. To date, only a limited number of Byzantine-robust DFL methods have been proposed, most of which are either communication-inefficient or remain vulnerable to advanced poisoning attacks. In this paper, we propose a new algorithm called BALANCE (Byzantine-robust averaging through local similarity in decentralization) to defend against poisoning attacks in DFL. In BALANCE, each client leverages its own local model as a similarity reference to determine if the received model is malicious or benign. We establish the theoretical convergence guarantee for BALANCE under poisoning attacks in both strongly convex and non-convex settings. Furthermore, the convergence rate of BALANCE under poisoning attacks matches those of the state-of-the-art counterparts in Byzantine-free settings. Extensive experiments also demonstrate that BALANCE outperforms existing DFL methods and effectively defends against poisoning attacks.

CCS CONCEPTS

• **Security and privacy** → **Systems security**.

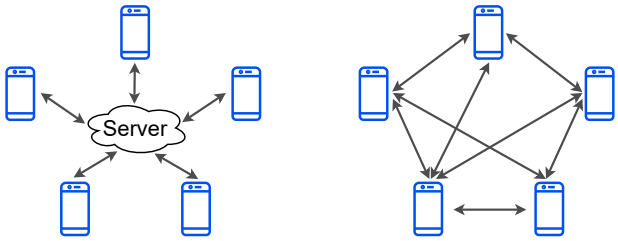
KEYWORDS

Decentralized Federated Learning, Poisoning Attacks, Byzantine Robustness

*Corresponding author

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10.
https://doi.org/10.1145/3658644.3670307



(a) Server-assisted FL (b) DFL

Figure 1: Server-assisted FL vs. DFL.

ACM Reference Format:
Minghong Fang, Zifan Zhang, Hairi, Prashant Khanduri, Jia Liu, Songtao Lu, Yuchen Liu, and Neil Gong. 2024. Byzantine-Robust Decentralized Federated Learning. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA*. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3658644.3670307

1 INTRODUCTION

Federated learning (FL) [35] has recently emerged as a powerful distributed learning paradigm that leverages multiple clients to train machine learning models collaboratively without sharing their raw training data. FL naturally follows the server-based distributed architecture, also known as *server-assisted federated learning (server-assisted FL)* [25, 35], where the training process is orchestrated by a server. However, despite its simplicity, the server-assisted FL framework suffers from *three key limitations* due to the reliance of a central server. The first limitation is that the server is vulnerable to the *single-point-of-failure risk*, which renders the server a clear target for cyber-attacks or the server itself could experience crashes or other system failures [13, 21, 22]. The second limitation of the server-assisted FL framework is that its single-level tree topology implies a *communication bottleneck* at the server (the root node) as the number of clients increases [13, 31, 45]. This communication bottleneck significantly worsens the scalability of large-scale distributed training over a server-based architecture. The third limitation is that server-assisted FL suffers from *trust dependency issues*: all participating clients have to trust the server, which has the potential to influence clients' models arbitrarily [17, 44, 45].

The limitations of these existing server-assisted FL systems have motivated researchers to pursue a fully-decentralized FL design, also known as *decentralized federated learning (DFL)* [4, 13, 14, 23,

28, 33, 45, 58]. In DFL, clients exchange information in a peer-to-peer fashion, without the assistance of a server. Clients in DFL follow the same process of training their local models as in server-assisted FL. However, in this fully-decentralized setting, each client only needs to send its updated model to its neighboring clients and perform local aggregation of the received models. Fig. 1 shows the difference between server-assisted FL and DFL. Thanks to its salient features, DFL has found a wide range of applications, such as healthcare services [34, 42, 47] and autonomous driving [4, 41].

Although DFL provides many benefits, a significant barrier to its widespread adoption is the susceptibility of DFL models to poisoning attacks. Malicious clients in FL could arbitrarily manipulate the system via poisoning their local training data (aka *data poisoning attacks*) [5, 38, 50] or local models (aka *model poisoning attacks*) [2, 3, 15, 30, 48, 53] to degrade the learning performance. To address this challenge, a number of Byzantine-robust FL algorithms have been proposed with varying degrees of success (e.g., [6, 8, 12, 18, 27, 37, 52, 56, 59]). However, almost all of these existing defenses are based on the server-assisted FL design. To defend against poisoning attacks in DFL systems, the straightforward approach is to adapt the existing defenses designed for server-assisted FL to DFL setting. However, our later experiments show that directly applying these server-based defenses to DFL leads to unsatisfactory performance since they are not designed for DFL architecture.

We note, however, that designing Byzantine-robust DFL algorithms is highly non-trivial. One of the main challenges is that, unlike server-assisted FL where the server maintains a single global model; in DFL, each client not only performs model training, but also acts as a "parameter server" to aggregate the received models. At the end of the training process, each client in DFL holds its own final trained model. The second challenge is that in DFL, clients connect to each other randomly, and different clients may have varying numbers of malicious neighbors. Furthermore, in DFL, clients interact exclusively with their topological neighbors. As a result, each client only has a "partial view" of the entire system. The above challenges make it difficult to guarantee that all benign clients in DFL obtain accurate final models, both theoretically and empirically. Recently, a few Byzantine-robust DFL methods have been proposed [14, 21, 22]. However, these DFL methods suffer from the following limitations: First, some approaches lack communication efficiency. For instance, in LEARN [14], each client needs to exchange information with its neighboring clients multiple times during each training round, resulting in a significant communication overhead. Second, certain defenses cannot provide theoretical guarantees that all benign clients will obtain accurate final models through the collaborative learning process. Moreover, even when such guarantees are provided, these methods need to assume that each benign client has knowledge of its malicious ratio (fraction of neighbors that are malicious). Third, as our experimental results will demonstrate, existing DFL methods are inherently vulnerable to poisoning attacks.

Our work: In this paper, we aim to bridge this gap. We propose a novel method called BALANCE (Byzantine-robust averaging through local similarity in decentralization) to defend against poisoning attacks in DFL. Our proposed BALANCE method is based on the observation that the attacker could manipulate the directions or magnitudes of local models on malicious clients in order to effectively corrupt the FL system. In our proposed method, each client uses its own local model as a *similarity reference* to assess whether the model it received is malicious or benign. The high-level idea of BALANCE is that if the received model is *close* to the client's own model in both direction and magnitude, it is considered benign; otherwise, the received model will be ignored. We provide theoretical guarantees of BALANCE under poisoning attacks in both strongly convex and non-convex settings. Specifically, in the case of a strongly convex population loss, we theoretically prove that for our BALANCE method, the final model learned by each benign client converges to the neighborhood of the global minimum. In the non-convex setting, we theoretically demonstrate that the final model of each benign client could converge to a neighborhood of a stationary point. Additionally, the convergence rates of our proposed method in both strongly convex and non-convex settings align with the optimal convergence rate observed in Byzantine-free strongly convex and non-convex optimizations, respectively. Notably, our theoretical guarantees are established without relying on the stringent and often unrealistic assumptions commonly made in existing DFL methods. These include the need for the communication graph to be complete and the requirement for each client to know the percentage of their neighbors who are malicious.

We extensively evaluate our proposed method on 5 datasets from different domains, 9 poisoning attacks (including attacks specifically developed for server-assisted FL and those customized for DFL architectures), 12 communication graphs, along with 8 state-of-the-art FL baselines. Furthermore, we explore various practical settings in DFL, including but not limited to, clients having highly non-independent and identically distributed training data (e.g., each client possessing data from merely three classes), clients employing different robust aggregation rules to combine the received models, clients starting with different initial models, various fractions of edges between malicious and benign clients, and time-varying communication graphs (e.g., clients may disconnect from the protocol due to Internet issues). We summarize our main contributions in this paper as follows:

- We propose BALANCE, a novel approach to defend against poisoning attacks in DFL. In contrast to existing DFL defenses, our BALANCE algorithm achieves the same communication complexity as that of the state-of-the-art server-assisted FL algorithms.
- We theoretically establish the convergence rate performance of BALANCE under poisoning attacks in both strongly convex and non-convex settings. We note that the convergence rate performance of BALANCE under strongly convex and non-convex settings match the optimal convergence rates in Byzantine-free strongly convex and non-convex optimizations, respectively.
- Our extensive experiments on different benchmark datasets, various poisoning attacks and practical DFL settings demonstrate and verify the efficacy of our proposed BALANCE method.

2 PRELIMINARIES AND RELATED WORK

Notations: Throughout this paper, matrices and vectors are denoted by boldface letters. We use $\|\cdot\|$ for ℓ_2-norm. For any given set $\mathcal{V}$, we use $|\mathcal{V}|$ denote its cardinality.

2.1 Decentralized Federated Learning (DFL)

Typically, in federated learning (FL), the training procedure can be formulated as an empirical risk minimization (ERM) problem. The aim is to learn a model $\boldsymbol{w}^*$ that minimizes the optimization problem expressed as follows:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}\in\Theta} F(\boldsymbol{w}) = \frac{1}{|D|}\sum_{\zeta\in D} f(\boldsymbol{w},\zeta), \tag{1}$$

where $\Theta \subset \mathbb{R}^d$ is the parameter space, d corresponds to the dimension of model parameter; $F(\cdot)$ denotes the population risk function; D denotes the entire training dataset; $f(\boldsymbol{w},\zeta)$ represents the empirical loss function, which is computed using weight parameter $\boldsymbol{w}$ and a training sample ζ.

In this paper, we try to solve the FL problem in (1) in a fully decentralized manner, without requiring assistance from a centralized server. Specifically, consider a DFL system with a set of clients $\mathcal{V}$. We let $|\mathcal{V}|$ denote the number of clients in the system. The network topology of this DFL system is defined by an undirected and unweighted communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E}$ denotes the set of edges between clients, and self-loops are not allowed. Communication is only possible between two clients if there is an edge connecting them. Each individual client, denoted as $i \in \mathcal{V}$, has its own private training dataset D_i. Collectively, we denote the joint global training dataset as $D = \bigcup_{i\in\mathcal{V}} D_i$. Every client i maintains a model $\boldsymbol{w}_i$ that is based on its local training data and information (e.g., model parameter in this paper) gathered from its neighboring clients. Specifically, in each training round t, each client conducts the following two steps:

Step I (Local model training and exchanging): Client $i \in \mathcal{V}$ performs local training to get an intermediate model $\boldsymbol{w}_i^{t+\frac{1}{2}}$, and subsequently sends $\boldsymbol{w}_i^{t+\frac{1}{2}}$ to its neighboring client $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is the set of neighbors of client i, *not* including the client i itself. At the same time, client i also receives the intermediate local model $\boldsymbol{w}_j^{t+\frac{1}{2}}$ from its neighboring client j. Lines 5-7 in Algorithm 2 summarize Step I. The local training of clients is shown in Algorithm 1.

Step II (Local model aggregation): Upon receiving all intermediate local models from its neighbors, each client $i \in \mathcal{V}$ aggregates and then updates its model as follows (Line 9 in Algorithm 2):

$$\boldsymbol{w}_i^{t+1} = \alpha \boldsymbol{w}_i^{t+\frac{1}{2}} + (1-\alpha)\mathrm{AGG}\{\boldsymbol{w}_j^{t+\frac{1}{2}}, j \in \mathcal{N}_i\}, \tag{2}$$

where α is a trade-off parameter, a larger α indicates trust more in the client's own intermediate local model, while a smaller value of α means put more weight on the aggregated neighboring models; $\mathrm{AGG}\{\boldsymbol{w}_j^{t+\frac{1}{2}}, j \in \mathcal{N}_i\}$ denotes that the client i employs a certain aggregation rule, represented by AGG, to combine the local models received from its neighboring clients. The aggregation rule AGG can be FedAvg [35] or Median [56].

In [45], clients update their local models as $\boldsymbol{w}_i^{t+1} = \mathrm{AGG}\{\boldsymbol{w}_j^{t+\frac{1}{2}}, j \in \widehat{\mathcal{N}_i}\}$, with $\widehat{\mathcal{N}_i} = \mathcal{N}_i \cup \{i\}$. Our experiments later reveal that even in non-adversarial settings, such aggregation leads to high error rates in final models using existing Byzantine-robust methods.

Algorithm 1 LocalTraining($\boldsymbol{w}, D, \eta$).

Output: $\boldsymbol{w}$.

1: **for** each local iteration **do**
2: Sample a mini-batch of training data from D to compute stochastic gradient $\boldsymbol{g}(\boldsymbol{w})$.
3: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\boldsymbol{g}(\boldsymbol{w})$.
4: **end for**

Algorithm 2 Training procedure of DFL.

Input: Set of clients $\mathcal{V}$; local training data D_i, $i \in \mathcal{V}$; training rounds T; learning rate η; communication graph $\mathcal{G}$; parameter α; aggregation rule AGG.

Output: Local models $\boldsymbol{w}_i^T$, $i \in \mathcal{V}$.

1: Initialize $\boldsymbol{w}_i^0$, $i \in \mathcal{V}$.
2: **for** $t = 0, 1, \cdots, T-1$ **do**
3: **for** each client $i \in \mathcal{V}$ in parallel **do**
4: // Step I: Local model training and exchanging.
5: $\boldsymbol{w}_i^{t+\frac{1}{2}} = \mathrm{LocalTraining}(\boldsymbol{w}_i^t, D_i, \eta)$.
6: Send $\boldsymbol{w}_i^{t+\frac{1}{2}}$ to all neighboring clients $j \in \mathcal{N}_i$.
7: Receive $\boldsymbol{w}_j^{t+\frac{1}{2}}$ from all neighboring clients $j \in \mathcal{N}_i$.
8: // Step II: Local model aggregation.
9: $\boldsymbol{w}_i^{t+1} = \alpha\boldsymbol{w}_i^{t+\frac{1}{2}} + (1-\alpha)\mathrm{AGG}\{\boldsymbol{w}_j^{t+\frac{1}{2}}, j \in \mathcal{N}_i\}$.
10: **end for**
11: **end for**

2.2 Poisoning Attacks to FL

FL is vulnerable to both data poisoning attacks [5, 38, 50] and model poisoning attacks [2, 3, 6, 9, 15, 19, 20, 49, 53, 57, 60]. In data poisoning attacks, malicious clients poison their training data. For instance, in a label flipping attack [50], the attacker flips the labels of local training data in malicious clients while leaving the features unchanged. Malicious clients can also manipulate their local models directly, which are known as model poisoning attacks [2, 6, 15, 20, 49]. Depending on the attacker's objective, model poisoning attacks can be categorized as either untargeted attacks [6, 15, 49] or targeted attacks [2, 20, 55]. In untargeted attacks, the attacker manipulates the FL system in a way that the final learned model will make incorrect predictions on a significant number of test examples without distinction. Conversely, in targeted attacks, the attacker seeks to influence the predictions of the final learned model on the specific inputs. A recent study [45] demonstrates that DFL is vulnerable to privacy attacks, this topic is out of the scope of this paper.

2.3 Byzantine-robust DFL Aggregation Rules

Since FL is vulnerable to poisoning attacks, many Byzantine-robust aggregation mechanisms for FL have been developed [6, 8, 10–12, 16, 18, 24, 27, 36, 39, 43, 54, 56]. However, most of them are based on the server-assisted FL design. Recently, a few Byzantine-robust FL methods have been proposed to tackle this challenge in DFL setting [14, 21, 22]. For instance, in the UBAR [21] method, each client first selects a group of neighboring clients that has the smallest sum of distances to its own local model, then further excludes information from neighbors that would result in a larger loss. LEARN [14] is another type of Byzantine-robust aggregation

Table 1: "Convex guarantee" and "Non-convex guarantee" mean the method provides theoretical performance guarantees under strongly convex and non-convex settings, respectively; "No know. about c_i" means benign client i has no knowledge about c_i (malicious ratio of client i); "No complete graph assum." means the approach does not require the assumption that the communication graph $\mathcal{G}$ must be a complete graph.; "No extra comm. cost" means the method does not incur extra communication cost compared to FedAvg.

Method	Convex guarantee	Non-convex guarantee	No know. about c_i	No complete graph assum.	No extra comm. cost
UBAR [21]	✗	✗	✗	✗	✓
LEARN [14]	✗	✓	✗	✗	✗
SCCLIP [22]	✗	✓	✓	✓	✓
BALANCE	✓	✓	✓	✓	✓

protocol designed for DFL. Clients in LEARN share both local model updates and local models with their neighboring clients in each training round. Specifically, each client first exchanges local model update with neighboring clients for $\lceil \log_2 t \rceil$ times, and then shares its local model one time, where t is the current training round. The trimmed mean aggregation rule is used by clients to combine the received local model updates and models. In the SCCLIP [22] aggregation rule, each client clips all received local models from neighboring clients to make sure the norm of a clipped received local model is no larger than that of the client's own model.

However, there are several inherent limitations in existing DFL defenses. First, UBAR cannot theoretically guarantee that every benign client could learn an accurate model. Second, although some methods offer theoretical guarantees for benign clients, they assume that each benign client i has knowledge of its malicious ratio c_i, which is computed as number of malicious neighbors divided by the total number of neighbors of client i. That is to say, in these methods, it is assumed that each benign client knows the number of neighbors that are malicious. Third, the LEARN method additionally presupposes that the underlying communication graph $\mathcal{G}$ must be a complete (fully connected) graph. Notably, the LEARN method incurs a large communication cost, as during each training round, clients need to exchange information with their peers several times. Contrary to existing DFL defenses, our proposed BALANCE method addresses the above limitations. We compare our proposed BALANCE with existing Byzantine-robust DFL methods, and summarize the comparison in Table 1. It is important to note that in Table 1, we do not compare our method with server-assisted FL methods such as Krum and Median. This is because the theoretical results of server-assisted FL methods cannot be straightforwardly transferred to the DFL framework, owing to significant variations in their architectures and operational procedures. DFL necessitates distinct theoretical frameworks and analyses tailored to its specific features and obstacles. Determining how to adapt the theoretical results of server-assisted FL methods to the DFL context is a challenging task and falls outside the scope of this paper.

3 PROBLEM STATEMENT

Threat Model: Similar to prior works [15, 21, 22, 48, 49], we assume that the attacker controls some malicious clients, those malicious clients could either poison their local training data or directly manipulate the local models that are sent to their neighboring clients. Note that each malicious client could only send malicious local models to its neighbors. Additionally, a malicious client could distribute different local models to different neighboring clients. We also remark that following [21, 22, 45], the attacker cannot change the communication graph $\mathcal{G}$ between clients. However, clients may disconnect from the DFL protocol in each training round because of Internet-related issues.

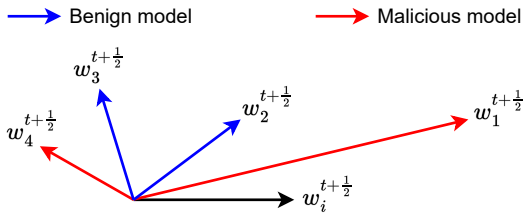


Figure 2: Illustration of our proposed BALANCE method.

Attacker's Knowledge: In terms of attacker's knowledge, following [8, 21, 22], we consider the worst-case attack scenario where the attacker has full-knowledge about the FL system, which includes local training data, the aggregation rule and trade-off parameter α utilized by all clients, as well as the communication graph $\mathcal{G}$. Note that in both non-adversarial and adversarial scenarios, each client knows the local models of its neighbors since local models are exchanged in DFL.

Defender's Knowledge and Goal: The proposed defense has no knowledge about the attacker's strategy nor the communication graph $\mathcal{G}$, but is expected to be capable of withstanding powerful adaptive attacks. It is important to note that in the proposed defense, each benign client is unaware of the total number of malicious clients in the system nor the number of neighboring clients that are malicious. We aim to develop a reliable and resilient DFL approach that meets the following three key goals. 1) Competitive learning performance: the proposed defense scheme for DFL should be effective in non-adversarial settings. Specifically, when there are no malicious clients, the model learned by each benign client using our proposed algorithm should attain comparable test error rate performance to that of averaging-based aggregation, which achieves state-of-the-art performance in non-adversarial settings; 2) Byzantine robustness: the proposed DFL method should demonstrate both theoretical and empirical resilience against Byzantine attacks; and 3) Communication and computation efficiency: the proposed algorithm should not result in any additional communication or computation costs when compared to FedAvg [35] in the absence of attacks.

4 THE BALANCE ALGORITHM

As summarized in Table 1, existing DFL methods either would incur a large communication cost, or make strong assumption that each benign client needs to know its malicious ratio. However, this assumption does not hold in practical scenarios, as in DFL, different clients connect to a different number of malicious clients. Moreover, ensuring that every benign client can learn an accurate final model by exchanging information with other clients presents a significant

challenge. In this section, we aim to design a simple yet effective DFL method to achieve three goals defined in Section 3.

As shown in [15, 48], the attacker could launch model poisoning attacks on FL either by manipulating the directions or magnitudes of the local models on malicious clients. In our proposed BALANCE, if the received intermediate model differs significantly from the client's own intermediate model, it is assumed to be potentially malicious and is ignored. Specifically, at training round t, when client $i \in \mathcal{V}$ receives a intermediate model $\boldsymbol{w}_j^{t+\frac{1}{2}}$ from its neighboring client $j \in \mathcal{N}_i$, it uses its own intermediate model $\boldsymbol{w}_i^{t+\frac{1}{2}}$ as a *similarity reference* to check whether the received model $\boldsymbol{w}_j^{t+\frac{1}{2}}$ is malicious or benign. If $\boldsymbol{w}_j^{t+\frac{1}{2}}$ is *close* to $\boldsymbol{w}_i^{t+\frac{1}{2}}$, both in terms of direction and magnitude, then client i will consider $\boldsymbol{w}_j^{t+\frac{1}{2}}$ as a benign model; otherwise client i will disregard $\boldsymbol{w}_j^{t+\frac{1}{2}}$. Additionally, as the model approaches convergence, $\boldsymbol{w}_j^{t+\frac{1}{2}}$ becomes more closer to $\boldsymbol{w}_i^{t+\frac{1}{2}}$. Based on the above insights, client i will only accept $\boldsymbol{w}_j^{t+\frac{1}{2}}$ if the following condition holds:

$$\|\boldsymbol{w}_i^{t+\frac{1}{2}} - \boldsymbol{w}_j^{t+\frac{1}{2}}\| \leq \gamma \cdot \exp(-\kappa \cdot \lambda(t)) \|\boldsymbol{w}_i^{t+\frac{1}{2}}\|, \tag{3}$$

where the parameter $\gamma > 0$ sets an upper limit for accepting a model as benign. The value of $\kappa > 0$ determines the rate at which the exponential function decreases; a larger κ results in a faster decay, while a smaller κ leads to a slower decay. The function $\lambda(t)$ is a monotonically increasing and non-negative function associated with the training round index t, meaning it becomes larger as t increases. When γ and κ are fixed, the term $\gamma \cdot \exp(-\kappa \cdot \lambda(t))$ decreases as t increases. Various methods exist for choosing $\lambda(t)$. For instance, a straightforward approach is to define it as $\lambda(t) = \frac{t}{T}$, where T is the total number of training rounds.

Fig. 2 shows the high-level idea of proposed BALANCE defense. In Fig. 2, $\boldsymbol{w}_1^{t+\frac{1}{2}}$, $\boldsymbol{w}_2^{t+\frac{1}{2}}$, $\boldsymbol{w}_3^{t+\frac{1}{2}}$, and $\boldsymbol{w}_4^{t+\frac{1}{2}}$ are four intermediate models sent from neighboring clients of client i; $\boldsymbol{w}_i^{t+\frac{1}{2}}$ is client i's own intermediate model. Client i will accept $\boldsymbol{w}_2^{t+\frac{1}{2}}$ and $\boldsymbol{w}_3^{t+\frac{1}{2}}$ because they are close to $\boldsymbol{w}_i^{t+\frac{1}{2}}$. However, models $\boldsymbol{w}_1^{t+\frac{1}{2}}$ and $\boldsymbol{w}_4^{t+\frac{1}{2}}$ are flagged as malicious since $\boldsymbol{w}_1^{t+\frac{1}{2}}$ deviates significantly from $\boldsymbol{w}_i^{t+\frac{1}{2}}$ in terms of magnitude, and $\boldsymbol{w}_4^{t+\frac{1}{2}}$ considerably differs from $\boldsymbol{w}_i^{t+\frac{1}{2}}$ in terms of direction.

During training round t, we define the set $\mathcal{S}_i^t \subseteq \mathcal{N}_i$ as the collection of neighboring clients of client i whose intermediate models satisfy Eq. (3). Client i then aggregates the received intermediate models from its neighboring clients by computing the average of all accepted models as $\frac{1}{|\mathcal{S}_i^t|} \sum_{j \in \mathcal{S}_i^t} \boldsymbol{w}_j^{t+\frac{1}{2}}$. Finally, client i could update its model by combining its own intermediate model with the aggregated intermediate model in the following manner:

$$\boldsymbol{w}_i^{t+1} = \alpha \boldsymbol{w}_i^{t+\frac{1}{2}} + (1-\alpha) \frac{1}{|\mathcal{S}_i^t|} \sum\nolimits_{j \in \mathcal{S}_i^t} \boldsymbol{w}_j^{t+\frac{1}{2}}. \tag{4}$$

Algorithm 3 shows the pseudocode of our proposed BALANCE algorithm. During the training round t, each client executes Lines 4-15 of Algorithm 3 in parallel. Specifically, for client $i \in \mathcal{V}$, it first performs local model training and exchanging (Lines 5-7). Note that the LocalTraining procedure is shown in Algorithm 1. If client i is a malicious client, it can choose to send arbitrary or carefully-crafted intermediate models to its neighboring clients at Line 6. After that, client i accepts intermediate models that satisfy Eq. (3) and further updates its local model (Lines 9-15).

Algorithm 3 BALANCE.

Input: Set of clients $\mathcal{V}$; local training data D_i, $i \in \mathcal{V}$; training rounds T; learning rate η; communication graph $\mathcal{G}$; parameters α, γ, κ and $\lambda(t)$.
Output: Local models $\boldsymbol{w}_i^T$, $i \in \mathcal{V}$.

1: Initialize $\boldsymbol{w}_i^0$, $i \in \mathcal{V}$.
2: **for** $t = 0, 1, \cdots, T-1$ **do**
3: **for** each client $i \in \mathcal{V}$ in parallel **do**
4: // Step I: Local model training and exchanging.
5: $\boldsymbol{w}_i^{t+\frac{1}{2}} = \text{LocalTraining}(\boldsymbol{w}_i^t, D_i, \eta)$.
6: Send $\boldsymbol{w}_i^{t+\frac{1}{2}}$ to all neighboring clients $j \in \mathcal{N}_i$.
7: Receive $\boldsymbol{w}_j^{t+\frac{1}{2}}$ from all neighboring clients $j \in \mathcal{N}_i$.
8: // Step II: Local model aggregation.
9: $\mathcal{S}_i^t = \emptyset$.
10: **for** each client $j \in \mathcal{N}_i$ **do**
11: **if** Eq. (3) satisfies **then**
12: $\mathcal{S}_i^t = \mathcal{S}_i^t \bigcup \{j\}$.
13: **end if**
14: **end for**
15: $\boldsymbol{w}_i^{t+1} = \alpha \boldsymbol{w}_i^{t+\frac{1}{2}} + (1-\alpha) \frac{1}{|\mathcal{S}_i^t|} \sum_{j \in \mathcal{S}_i^t} \boldsymbol{w}_j^{t+\frac{1}{2}}$.
16: **end for**
17: **end for**

Complexity analysis: In our proposed BALANCE method, at training round t, client $i \in \mathcal{V}$ computes the distance between its own intermediate model $\boldsymbol{w}_i^{t+\frac{1}{2}}$ and the received intermediate model $\boldsymbol{w}_j^{t+\frac{1}{2}}$ from a neighboring client $j \in \mathcal{N}_i$. Since the dimension of local model is d, and client i has $|\mathcal{N}_i|$ neighboring clients, the computational complexity of each client in our method is $O(d|\mathcal{N}_i|)$ at each training round.

5 THEORETICAL PERFORMANCE ANALYSIS

In this section, we present the convergence performance guarantee of our proposed BALANCE. We let $\mathcal{B} \subseteq \mathcal{V}$ be the set of benign clients. Let $\mathcal{G}_B$ be the subgraph induced by benign clients. In a convex setting, we denote the global minimum as $\boldsymbol{w}^*$; while in a non-convex setting, $\boldsymbol{w}^*$ represents a stationary point (a point which has zero gradient). Before introducing the theoretical results, we first present some technical assumptions that are standard in the literature [14, 22, 24, 31, 56].

Assumption 1. *The population risk $F(\boldsymbol{w})$ is μ-strongly convex, i.e., for all $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \Theta$, one has that:*

$$F(\boldsymbol{w}_1) + \langle \nabla F(\boldsymbol{w}_1), \boldsymbol{w}_2 - \boldsymbol{w}_1 \rangle + \frac{\mu}{2} \|\boldsymbol{w}_2 - \boldsymbol{w}_1\|^2 \leq F(\boldsymbol{w}_2).$$

Assumption 2. *The population risk $F(\boldsymbol{w})$ is L-smooth, i.e., for all $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \Theta$, we have that:*

$$\|\nabla F(\boldsymbol{w}_1) - \nabla F(\boldsymbol{w}_2)\| \leq L \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|.$$

Assumption 3. *The stochastic gradient $\boldsymbol{g}(\boldsymbol{w}_i)$ computed by a benign client $i \in \mathcal{B}$ is an unbiased estimator of the true gradient, and $\boldsymbol{g}(\boldsymbol{w}_i)$ has bounded variance, where $\mathcal{B}$ is the set of benign clients. That is, $\forall i \in \mathcal{B}$, one has that:*

$$\mathbb{E}[\boldsymbol{g}(\boldsymbol{w}_i)] = \nabla F(\boldsymbol{w}_i), \quad \mathbb{E}[\|\boldsymbol{g}(\boldsymbol{w}_i) - \nabla F(\boldsymbol{w}_i)\|]^2 \leq \delta^2.$$

Assumption 4. *For any benign client $i \in \mathcal{B}$, the model $\boldsymbol{w}_i$ and $\|\nabla F(\boldsymbol{w}_i)\|$ are bounded. That is, $\forall i \in \mathcal{B}$, we have $\|\boldsymbol{w}_i\| \leq \psi$, and $\|\nabla F(\boldsymbol{w}_i)\| \leq \rho$.*

Assumption 5. *$\mathcal{G}_B$ is connected.*

With the above assumptions, we provide the theoretical findings of our BALANCE both in strongly convex and non-convex settings. In the strongly convex setting, we have the following result.

Theorem 1 (The Strongly Convex Setting). *Suppose Assumptions 1-5 hold, clients select intermediate models according to Eq. (3). Let the learning rate η and γ be chosen as such that $\eta \leq \min\{\frac{1}{4L}, \frac{1}{\mu}\}$ and $\gamma \leq \frac{\rho}{L\psi(1-\alpha)}$. The value of $\kappa \cdot \lambda(t)$ is larger than 0. After T training rounds, for any benign client $i \in \mathcal{B}$, it holds that:*

$$\mathbb{E}[F(\boldsymbol{w}_i^T) - F(\boldsymbol{w}^*)] \leq (1-\mu\eta)^T [F(\boldsymbol{w}_i^0) - F(\boldsymbol{w}^*)] + \frac{2L\eta\delta^2}{\mu} + \frac{2\gamma\rho\psi(1-\alpha)}{\mu\eta},$$

where $\boldsymbol{w}_i^0$ is client i's initial model.

Proof. The proof is relegated to Appendix A. □

Theorem 1 says that by choosing an appropriate learning rate η and γ, for any benign client, the final learned model converges to the neighborhood of the global minimum. More importantly, the linear convergence rate is the *same* as the optimal convergence rate in Byzantine-free strongly convex optimization algorithm. In other words, Byzantine attacks do not hurt the convergence rate of our proposed method. Note that in this paper, we only consider standard gradient descent, without using the higher-order information or relying on accelerated methods.

In the non-convex setting, we have the following result.

Theorem 2 (The Non-convex Setting). *Under Assumptions 2-5, clients select intermediate models according to Eq. (3). Select η and γ such that $\eta \leq \min\{\frac{1}{4L}, \frac{1}{\mu}\}$ and $\gamma \leq \frac{\rho}{L\psi(1-\alpha)}$. In addition, $\kappa \cdot \lambda(t) > 0$. After T training rounds, the following holds for any benign client $i \in \mathcal{B}$:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\|\nabla F(\boldsymbol{w}_i^t)\|^2] \leq \frac{2[F(\boldsymbol{w}_i^0) - F(\boldsymbol{w}^*)]}{\eta T} + 4L\eta\delta^2 + \frac{4\gamma\rho\psi(1-\alpha)}{\eta}.$$

Proof. The proof is relegated to Appendix B. □

Theorem 2 shows that by selecting suitable parameters, the final model of each benign client could converge to the neighborhood of a stationary point. The sub-linear convergence rate aligns with the optimal convergence rate in a Byzantine-free non-convex optimization algorithm. In other words, poisoning attacks do not impact the convergence rate of our BALANCE in the non-convex setting.

Remark. *Assumption 3 states that the training data among clients are independent and identically distributed (IID), but this is not required for our experiments. In Assumption 5, it is posited that the subgraph $\mathcal{G}_B$, which is formed by benign clients, remains connected after eliminating all malicious clients and their corresponding edges. This assumption is critical, as it prevents the scenario where a benign client is exclusively surrounded by malicious neighbors. Our BALANCE does not require benign clients to be aware of the architecture of communication graph $\mathcal{G}$. Additionally, in our theoretical analysis, there is no necessity for $\mathcal{G}$ to have a particular architecture, such as the requirement for it to be a complete graph, as assumed in [14].*

Remark. *For a strongly-convex (or convex) objective, our Theorem 1 guarantees the convergence to a global optimal solution. For a non-convex objective, since guaranteeing convergence to a global optimal is NP-Hard [40], guaranteeing convergence to a stationary point (local optimal) is the best one can hope for in the non-convex case. The convergence rates of our BALANCE method in both strongly convex and non-convex scenarios match the best-known convergence rates of their Byzantine-free counterparts. We also note that using our proposed BALANCE does not require knowing the precise values of certain parameters introduced in assumptions, such as δ, ψ, and ρ. Clients only need to check whether Eq. (3) is met while filtering out malicious local models. Since $\gamma \cdot \exp(-\kappa \cdot \lambda(t))$ is always upper bounded by γ, Theorem 1 and Theorem 2 rely on the condition that the value of γ is bounded.*

6 EXPERIMENTS

6.1 Experimental Setup

6.1.1 Datasets and Poisoning Attacks. In our experiment, we assess our method and various baselines across multiple datasets, including a synthetic dataset and four real-world datasets: MNIST [29], Fashion-MNIST [51], Human Activity Recognition (HAR) [1], and CelebA [32]. Notably, HAR, sourced from 30 smartphone users (each representing a client), exemplifies a real-world FL dataset. Details on the creation of the synthetic dataset and specifics of the other four datasets are available in Appendix D.1.

We first consider seven poisoning attacks, including two data poisoning attacks (Label flipping (LF) attack [50], Feature attack) and five model poisoning attacks (Gaussian (Gauss) attack [6], Krum attack [15], Trim attack [15], Backdoor attack [2, 20], and Adaptive (Adapt) attack [48]). Note that Backdoor attack is a targeted attack model, where the attacker aims to craft the system such that the final trained model makes incorrect predictions on inputs selected by the attacker. Adapt attack is the most powerful attack, where the attacker has full knowledge of the system, including all benign clients' local models and the proposed aggregation rule BALANCE used by clients. The attacker in Adapt attack introduces minor perturbation to the benign local models to create malicious models. The detailed description of seven poisoning attacks is shown in Appendix D.2. Additionally, we evaluate two other attacks in Section 7: "a little is enough" (LIE) attack [3], and the Dissensus attack [22], a new form of attack specifically designed for DFL systems.

6.1.2 Comparison DFL Methods. We evaluate the effectiveness of our proposed BALANCE by comparing it with the following eight

methods. Note that FedAvg [35], Krum [6], Trimmed Mean (Trimmean) [56], Median [56], and FLTrust [8] were originally designed for server-assisted FL, which are adapted to the DFL setting.

FedAvg [35]: In the FedAvg method, a client collects local models from its neighbor clients, then takes the weighted average of all collected models.

Krum [6]: When client $i \in \mathcal{V}$ gets $|\mathcal{N}_i|$ local models from neighbors, it chooses the model closest in Euclidean distance to its $|\mathcal{N}_i| - \lceil c_i|\mathcal{N}_i|\rceil - 2$ nearest models. Here, $\mathcal{N}_i$ is client i's neighbor set, with c_i and $\lceil c_i|\mathcal{N}_i|\rceil$ denoting the proportion and count of malicious neighbors, respectively.

Trimmed Mean (Trim-mean) [56]: Once client $i \in \mathcal{V}$ receives $|\mathcal{N}_i|$ local models from its neighbors, it first removes the largest and smallest $\lceil c_i|\mathcal{N}_i|\rceil$ elements for each dimension, then computes the average of the rest.

Median [56]: In the Median rule, each client i computes the coordinate-wise median of all collected $|\mathcal{N}_i|$ local models.

FLTrust [8]: In FLTrust, client i calculates the cosine similarity between its local model $\boldsymbol{w}_i^{t+\frac{1}{2}}$ and a neighbor's model $\boldsymbol{w}_j^{t+\frac{1}{2}}$ upon receipt. If this similarity is positive, $\boldsymbol{w}_j^{t+\frac{1}{2}}$ is normalized to $\tilde{\boldsymbol{w}}_j^{t+\frac{1}{2}}$ with the same magnitude as $\boldsymbol{w}_i^{t+\frac{1}{2}}$, followed by client i averaging all normalized models received from neighbors.

UBAR [21]: The UBAR aggregation rule employs a two-stage process to filter out any potentially malicious information. Specifically, during training round t, client i first identifies a subset of neighbors $\mathcal{N}_i^s$, which consists of those with the smallest sum of distance to $\boldsymbol{w}_i^{t+\frac{1}{2}}$, where $\mathcal{N}_i^s \subseteq \mathcal{N}_i$, $|\mathcal{N}_i^s| = |\mathcal{N}_i| - \lceil c_i|\mathcal{N}_i|\rceil$. In the second stage, client i narrows down the subset even further by selecting a new subset $\mathcal{N}_i^r$ from $\mathcal{N}_i^s$, which only includes neighbors whose loss values are smaller than its own loss. Finally, client i averages the local models from $\mathcal{N}_i^r$.

LEARN [14]: In LEARN, clients exchange both local model updates and local models with their neighboring clients, and utilize Trimmean aggregation rule to combine the received local model updates and local models. Specifically, during training round t, client i aggregates local model updates from its neighboring clients for $\lceil \log_2 t \rceil$ times, then exchanges local models with its neighbors once.

Self-Centered Clipping (SCCLIP) [22]: In the SCCLIP aggregation rule, each client clips all received local models from its neighbor clients based on its own local model.

6.1.3 Evaluation Metrics. For the synthetic dataset, we employ maximum mean squared error (Max.MSE) as the evaluation criterion, as we train a linear regression model on this synthetic dataset. For the four real-world datasets, we use maximum testing error rate (Max.TER) and maximum attack success rate (Max.ASR) as the evaluation metrics, as these datasets are used for training classification models. For all three evaluation metrics, smaller values indicate stronger defense capabilities.

Maximum mean squared error (Max.MSE): In the linear regression model, we first calculate the mean squared error (MSE) for each benign client's final local model. The MSE is computed as $\text{MSE} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2$, where y_i is the actual value, $\hat{y}_i$ denotes the predicted value, and n_{test} is the number of testing examples. We then assess a DFL method's robustness on the synthetic dataset by selecting the maximum MSE among all benign clients.

Maximum testing error rate (Max.TER) [21]: Following [21], we compute the testing error rate of the final local model on each benign client, and use the maximum testing error rate among all benign clients to measure the robustness of a DFL method.

Maximum attack success rate (Max.ASR): We compute the attack success rate of the final local model on each benign client, and report the maximum attack success rate among all benign clients. The attack success rate is the fraction of targeted testing examples classified as the attacker-chosen targeted label.

6.1.4 Non-IID Setting. Training data in FL are typically Non-IID (not independently and identically distributed) across clients. In our paper, we consider the IID setting for synthetic dataset, and Non-IID setting for four real-world datasets. We use the way in [15] to simulate the Non-IID setting for MNIST, Fashion-MNIST datasets. In this approach, for a dataset containing z classes, clients are first divided into z random groups. A training example labeled h is allocated to clients in group h with a specific probability p, and to those in different groups with a probability of $\frac{1-p}{z-1}$. Within the same group, the training examples are evenly distributed among the clients. An increase in p results in a greater level of Non-IID. In our experiment, we set $p = 0.8$, indicating a substantial imbalance in the distribution of labels among clients. For example, 80% of the training data for a client is concentrated in a single class. In Section 7, we explore a more extreme Non-IID scenario where each client's training data is limited to just a few classes (e.g., three). The HAR dataset's training data are inherently heterogeneous, eliminating the need for Non-IID simulation. Similarly, the CelebA dataset, processed as per [7], already exhibits Non-IID characteristics, so additional Non-IID simulation is unnecessary.

6.1.5 Parameter Setting. We assume that there are a total of 20 clients for synthetic, MNIST, Fashion-MNIST, and CelebA datasets. Note that each smartphone user can be seen as a client in the HAR dataset, thus there are 30 clients in total for that dataset. By default, we assume that 20% of clients are malicious. In our experiments, we train a linear regression model on the synthetic dataset. Note that the population risk of the linear regression model satisfies Assumption 1 and Assumption 2. We use a convolutional neural network (CNN) to train the MNIST, Fashion-MNIST, and CelebA datasets. The architecture of CNN is shown in Table 7 in Appendix. For the HAR dataset, we train a logistic regression classifier. We train 300, 2,000, 2,000, 1,000 and 1,500 rounds for synthetic, MNIST, Fashion-MNIST, HAR and CelebA datasets, respectively. The learning rates are respectively set to 6×10^{-4}, 3×10^{-4}, 6×10^{-3}, 3×10^{-3} and 5×10^{-5} for five datasets. For all datasets, we set $\alpha = 0.5$, $\gamma = 0.3, \kappa = 1, \lambda(t) = \frac{t}{T}$.

By default, we assume that all clients use the same initial local model, parameters $\alpha, \gamma, \kappa, \lambda(t)$, and aggregation rule AGG. We will also explore the settings where clients have different initial local models, α, and AGG. Aligned with existing work [45], we consider regular graph as the default communication graph, where each node has an equal number of neighboring nodes. We use regular-(n, v) to denote a regular graph with n nodes, where each node is connected

to v neighbors. By default, we use a regular-(20, 10) graph for the synthetic, MNIST, Fashion-MNIST, and CelebA datasets. The HAR dataset inherently consists of 30 clients, so we consider a regular-(30, 15) graph structure for HAR. Fig. 9a and Fig. 9b in Appendix show the topologies of regular-(20, 10) and regular-(30, 15) graphs, respectively. Note that in Fig. 9, each node represents a client. The nodes highlighted in red indicate malicious clients, while the nodes highlighted in blue represent benign clients. By default, we assume that the communication graph $\mathcal{G}$ is static, i.e., the edges between clients will not change over time. We will also explore the time-varying communication graph setting, where each client has certain possibility of not sharing information with its neighboring clients in each round. We perform experiments on four NVIDIA Tesla V100 GPUs, repeating each experiment 10 times and averaging the results. Default results are reported for the MNIST dataset using a regular-(20, 10) graph, with 4 out of 20 clients being malicious.

Table 2: Results of different DFL methods. The results of Backdoor are presented as "Max.TER / Max.ASR".

(a) MNIST dataset.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.10	0.90	0.90	0.91	0.91	0.90 / 1.00	0.90
Krum	0.10	0.12	0.90	0.10	0.10	0.15	0.15 / 0.01	0.14
Trim-mean	0.11	0.12	0.49	0.11	0.82	0.81	0.83 / 0.72	0.87
Median	0.14	0.14	0.45	0.15	0.52	0.63	0.66 / 0.01	0.66
FLTrust	0.10	0.11	0.90	0.13	0.10	0.88	0.10 / 0.73	0.10
UBAR	0.14	0.14	0.90	0.14	0.14	0.14	0.15 / 0.01	0.14
LEARN	0.10	0.10	0.30	0.12	0.18	0.57	0.12 / 0.03	0.44
SCCLIP	0.10	0.10	0.10	0.11	0.91	0.91	0.10 / 0.01	0.91
BALANCE	0.10	0.10	0.11	0.10	0.10	0.11	0.11 / 0.01	0.11

(b) Fashion-MNIST dataset.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.16	0.21	0.90	0.90	0.90	0.90	0.90 / 1.00	0.90
Krum	0.26	0.27	0.90	0.40	0.37	0.48	0.28 / 0.03	0.42
Trim-mean	0.25	0.27	0.90	0.27	0.77	0.87	0.90 / 1.00	0.76
Median	0.26	0.26	0.90	0.28	0.54	0.74	0.90 / 1.00	0.69
FLTrust	0.19	0.20	0.19	0.90	0.25	0.90	0.19 / 0.99	0.90
UBAR	0.21	0.23	0.90	0.21	0.22	0.22	0.24 / 0.03	0.23
LEARN	0.23	0.26	0.47	0.23	0.34	0.37	0.23 / 0.90	0.51
SCCLIP	0.20	0.25	0.90	0.33	0.89	0.89	0.90 / 1.00	0.52
BALANCE	0.16	0.17	0.17	0.16	0.17	0.17	0.17 / 0.02	0.17

(c) HAR dataset.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.04	0.04	0.45	0.98	0.32	0.38	0.82 / 1.00	0.99
Krum	0.10	0.10	0.10	0.10	0.10	0.10	0.10 / 0.01	0.10
Trim-mean	0.05	0.06	0.06	0.06	0.09	0.17	0.08 / 0.01	0.09
Median	0.06	0.06	0.08	0.06	0.07	0.17	0.07 / 0.01	0.08
FLTrust	0.04	0.04	0.08	0.07	0.04	0.31	0.04 / 0.47	0.05
UBAR	0.06	0.06	0.06	0.06	0.06	0.12	0.08 / 0.03	0.06
LEARN	0.04	0.04	0.04	0.04	0.06	0.13	0.05 / 0.04	0.06
SCCLIP	0.05	0.05	0.13	0.07	0.27	0.32	0.06 / 0.02	0.12
BALANCE	0.04	0.05	0.04	0.05	0.04	0.05	0.04 / 0.01	0.05

(d) CelebA dataset.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.16	0.48	0.48	0.53	0.53	0.48 / 0.01	0.48
Krum	0.18	0.26	0.48	0.30	0.31	0.18	0.20 / 0.26	0.18
Trim-mean	0.12	0.24	0.35	0.15	0.15	0.26	0.22 / 0.09	0.19
Median	0.13	0.17	0.31	0.15	0.15	0.26	0.21 / 0.15	0.19
FLTrust	0.10	0.14	0.10	0.11	0.11	0.53	0.12 / 0.06	0.10
UBAR	0.12	0.13	0.48	0.13	0.12	0.14	0.14 / 0.13	0.13
LEARN	0.31	0.41	0.37	0.35	0.51	0.53	0.32 / 0.09	0.53
SCCLIP	0.10	0.17	0.14	0.12	0.43	0.53	0.48 / 0.01	0.53
BALANCE	0.10	0.12	0.11	0.11	0.11	0.11	0.12 / 0.02	0.13

6.2 Experimental Results

Our proposed BALANCE is effective: We first demonstrate the effectiveness of our proposed BALANCE on synthetic dataset, where the population risk is both μ-strongly and L-smooth, i.e., satisfying Assumption 1 and Assumption 2. Table 8 in Appendix shows the results of different methods under different attacks on synthetic dataset. Each row corresponds to a different DFL method. "No" means all clients are benign. We exclude Backdoor attacks for the synthetic dataset as there are no specific Backdoor attacks for regression models. From Table 8, we observe that our proposed method outperforms baselines in both non-adversarial and adversarial scenarios, the Max.MSEs of our method are comparable to those of FedAvg without attacks.

Next, we show the performance of our method on four real-world datasets, where the trained models are highly non-convex, results are shown in Table 2. The results for the Backdoor attack are given as"Max.TER / Max.ASR". We note that DFL method achieves comparable performance to its server-assisted counterpart. For instance, when FedAvg aggregation rule is used and all clients are benign, the test error of the final global model is 0.09 in server-assisted FL. We also remark that in DFL, clients could not obtain accurate final models when they independently train their models without exchanging information with other clients. On MNIST dataset, the Max.TER is 0.29 when clients solely train models locally.

First, we observe that when there is no attack, i.e., all clients are benign, our proposed BALANCE achieves similar Max.TER as that of FedAvg under no attack. This means that our method achieves the "competitive learning performance" goal mentioned in Section 3. For instance, on the CelebA dataset, both our proposed method and FedAvg under no attack exhibit a Max.TER of 0.10. However, the Max.TER of LEARN is 0.31, see Table 2d. Next, we find that our proposed BALANCE is resilient to different types of poisoning attacks, including data poisoning and model poisoning attacks, and performs better than existing methods. For instance, on the MNIST dataset, Trim-mean's Max.TER increases from 0.11 to 0.81 under the Trim attack. In contrast, our method maintains a small corresponding Max.TER of 0.11. We observe similar results on the other three real-world datasets, indicating that our BALANCE achieves the "Byzantine robustness" goal. We remark that BALANCE either matches or outperforms all existing methods known to converge to (global) optimal points. This shows BALANCE does not get stuck at non- or local-optimal points. We also note that the Adapt attack demonstrates the most effective attack performance when targeting our proposed method, whereas it may perform worse when attacking other methods. The reason is that the Adapt attack is specifically designed for our method.

Fig. 3 shows the testing error rate of each benign client, when clients utilize FedAvg without any attacks, and when they use Trim-mean aggregation rule and our proposed BALANCE under Trim attack. We observe that under Trim attack, the testing error rate of each benign client's final learned model escalates substantially when the clients adopt the Trim-mean aggregation rule to merge the local models from neighboring clients. However, our proposed method guarantees that each benign client will obtain a final model that is almost as accurate as FedAvg without any attacks.

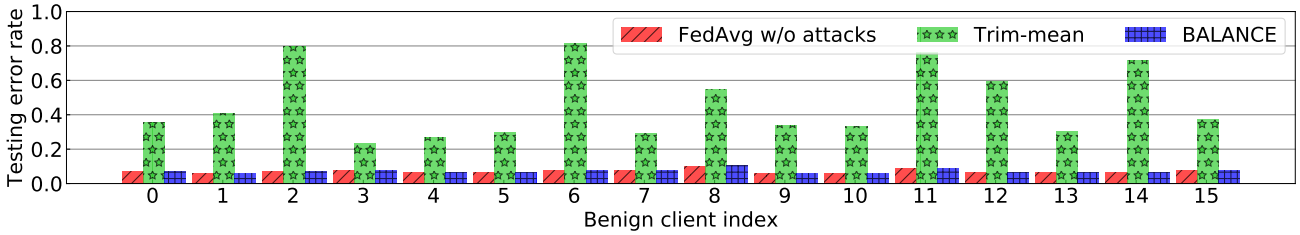
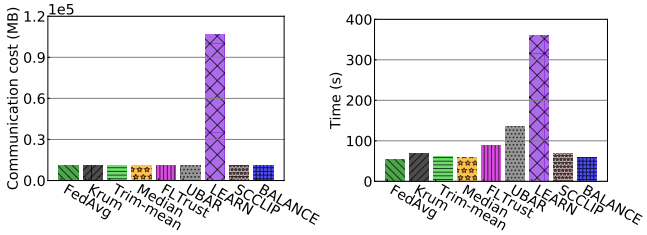
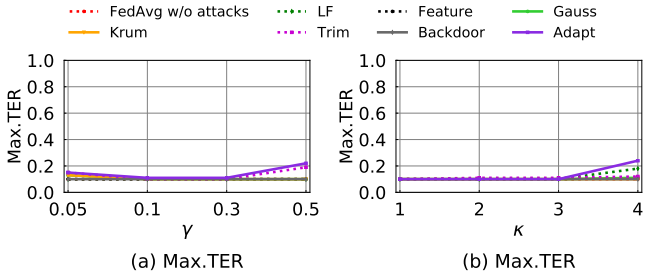


Figure 3: Testing error of each benign client of FedAvg without any attacks, Trim-mean aggregation rule and our proposed method under Trim attack.



(a) Communication cost. **(b) Computation cost.**

Figure 4: Communication and computation costs of different methods.



(a) Max.TER (b) Max.TER

Figure 5: Impact of γ and κ.

Fig. 4 shows the communication and computation costs of various methods when we train the FL system for 2,000 rounds on the MNIST dataset, where regular-(20, 10) communication graph is used. More specifically, for a given FL method, the communication cost refers to the size of data (local model or local model update) that each client sends to its neighboring clients over 2,000 rounds, while the computation cost indicates the time that each client requires to aggregate the received local models (updates) over 2,000 rounds. As seen in Fig. 4, our BALANCE demonstrates both communication and computation efficiency. Conversely, other methods lead to high communication and computation costs. For example, in each training round of the LEARN method, each client must first exchange local model updates with its neighboring clients for $\lceil \log_2 t \rceil$ rounds, and then exchanges local model once. This information exchange process incurs significant communication and computation costs.

From Table 2 and Table 8, we also observe that the application of server-assisted FL methods to DFL results in suboptimal performance. Specifically, FLTrust is particularly prone to Trim attack on the MNIST dataset. This vulnerability arises from FLTrust's underlying assumption that the server's root dataset mirrors the distribution of the clients' overall training data, an assumption that often does not hold in practical FL scenarios. Moreover, the training data of clients in DFL are highly heterogeneous. As a result, when a client employs FLTrust for aggregation, it tends to incorrectly classify many benign neighboring clients as malicious.

Table 9 in Appendix presents consensus errors [22, 26, 31] for various methods. Consensus error measures the average squared difference between each benign client's final model and all benign clients' average model. The details of this metric is shown in Section D.3 in Appendix. Our method shows low values in this metric. We also remark that consensus error alone cannot determine whether the final learned model is accurate or not. A small consensus error may also imply that all benign clients have reached a poor consensus, meaning that all benign clients have learned similar but inaccurate models. Thus we omit the consensus error metric in subsequent experiments. Note that we also do not consider the average testing error rate (Avg.TER) of benign clients, since low Avg.TER can sometimes mask high errors in individual clients.

Impact of γ and κ: Fig. 5 shows the results of our proposed BALANCE under various poisoning attacks with different values of γ and κ. We observe that the Max.TER of BALANCE is large when γ and κ are too large. The reason is that for our method, a client would falsely reject local models shared by benign neighboring clients when κ is too large, as a large κ leads to a rapid decrease in $\gamma \exp(-\kappa \cdot \lambda(t))$. The local models from malicious neighboring clients may get accepted if γ is too large.

Impact of fraction of malicious clients: Fig. 6 shows the results of different methods under different attacks on MNIST dataset and regular-(20, 10) communication graph, when the fraction of malicious clients varies from 0% to 80%, and the total number clients is set to 20. We observe that our proposed DFL approach is the only method that can withstand 50% of malicious clients, while existing Byzantine-robust methods lead to significant Max.TERs even when only a small proportion of clients are malicious. For example, UBAR aggregation rule is susceptible to poisoning attacks when only 10% of clients are malicious, as seen in the case of the Feature attack strategy, where the maximum testing error rate rises to 0.90. Furthermore, our proposed approach can withstand even the most powerful Adapt attack when 80% of clients are malicious.

Impact of degree of Non-IID: Fig. 7 displays the results of different methods under poisoning attacks with varying degrees of Non-IID. We observe that our proposed method outperforms existing DFL methods with all considered Non-IID scenarios. For example, when the degree of Non-IID is relatively low, such as 0.5, the Trim attack on the Median aggregation rule leads to a Max.TER of 0.32. However, for our proposed method, the Max.TERs of all benign

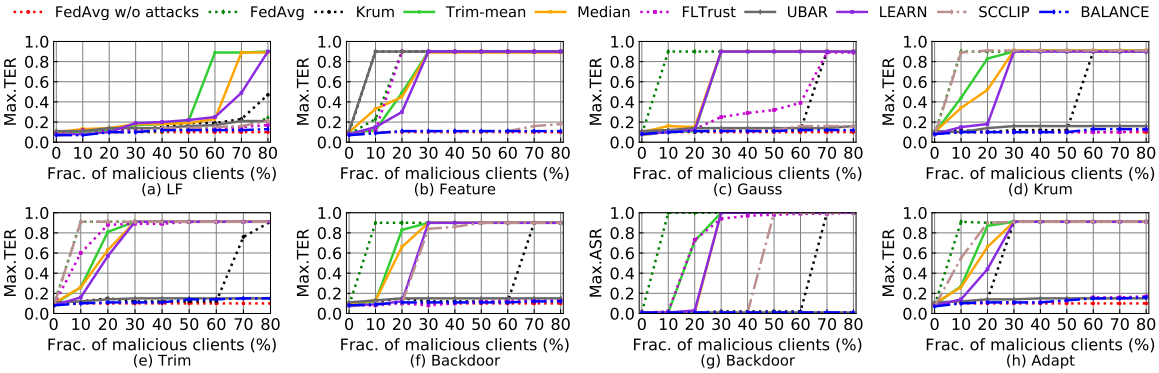


Figure 6: Impact of fraction of malicious clients.

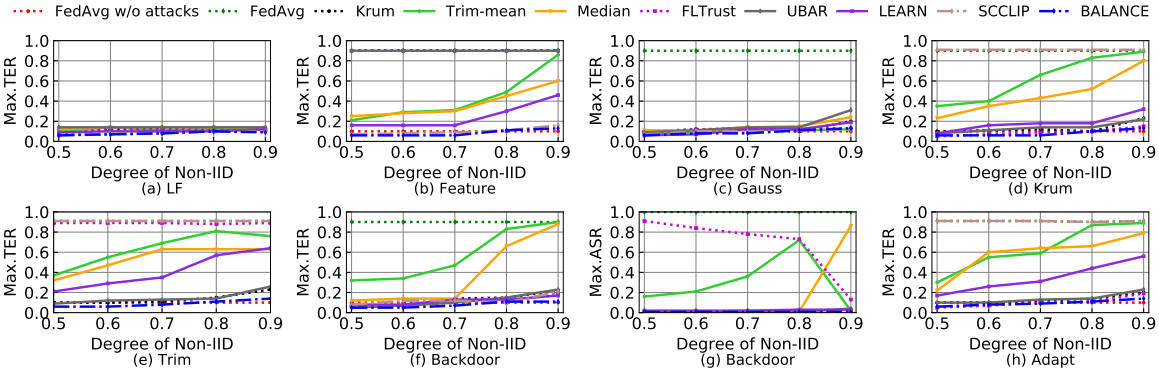


Figure 7: Impact of degree of Non-IID.

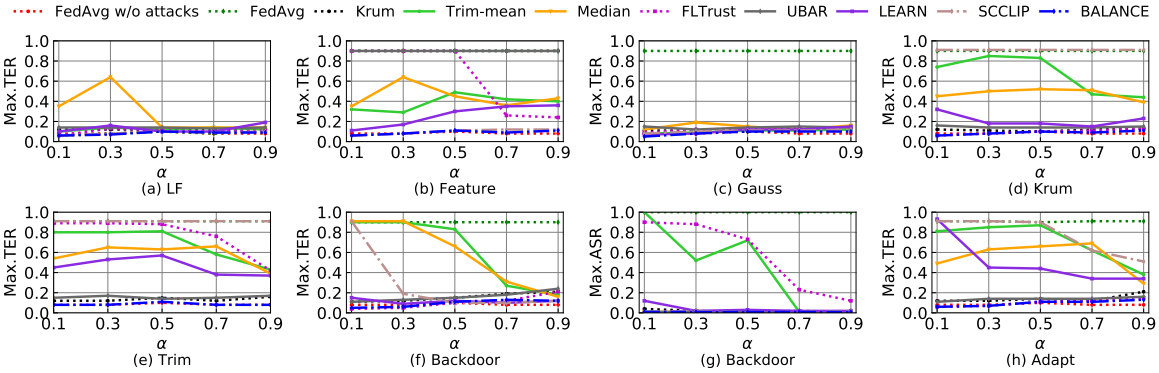


Figure 8: Impact of α.

clients are not significantly high, even with highly heterogeneous training data across all clients.

Impact of α: According to Eq. (2), in DFL, each client utilizes a parameter called α to balance the combination of its local model and those of its neighboring clients. A higher value of α indicates greater trust in the client's own local model, while a smaller value of α results in more weight placed on the aggregated neighboring models. Note that by default, we assume that all clients use the same α. Fig. 8 shows the results of different methods under various poisoning attacks, when we vary the value of α, where other parameters are set to their default settings. We can observe that in a particular DFL method, when the poisoning attack is weak, clients can achieve greater model accuracy by setting a smaller value of the trade-off parameter α. This means that clients can benefit more by collaborating with others and giving more weight to the models received from their neighboring clients. For example, for our proposed BALANCE under Gauss attack, the Max.TERs are 0.05 and 0.10 when α is set to 0.1 and 0.9, respectively. However, when the attack is strong, a smaller value of α may result in a larger Max.TER. This is because benign clients may receive malicious models from their neighboring clients, and if the DFL method is not robust, giving more weight to the neighboring models through a smaller α may lead to a larger Max.TER. For example, under Trim attack, when $\alpha = 0.1$, the Max.TER of the Trim-mean method is 0.80, while when α is set to 0.9, the Max.TER is 0.43. Note that in the extreme case where $\alpha = 1$, which means each client trains its

own local model independently without sharing information with others. In our experiment, we find that the Max.TER is 0.29 when each client only uses its own local training data to train the model.

The paper [45] employs a model aggregation technique where each client aggregates its model as $\boldsymbol{w}_i^{t+1} = \text{AGG}\{\boldsymbol{w}_j^{t+\frac{1}{2}}, j \in \widehat{\mathcal{N}_i}\}$, where $\widehat{\mathcal{N}_i} = \mathcal{N}_i \cup \{i\}$, $\mathcal{N}_i$ is the set of neighbors of client i (not including client i itself). Our proposed method is compared to existing DFL methods using this aggregation setting, and the results are shown in Table 10 in Appendix. We observe that our proposed method is also robust against various poisoning attacks and outperforms baseline methods under this setting. We also observe that if clients aggregate their models using the setting suggested in [45], the Max.TERs of existing defenses without attacks are very large. For instance, the Max.TERs of Krum and UBAR are respectively 0.18 and 0.25 when all clients are benign, however, the corresponding Max.TERs are 0.10 and 0.14 (see the results in Table 2a), respectively when clients perform aggregation based on Eq. (2).

Clients use different α or different aggregation rules: By default, in our experiments, clients use the same α and follow the same aggregation rule to combine their local models with their neighboring clients' models. In this section, we first investigate the scenario where different clients use different α values. In our experiments, each client randomly samples its α value from the interval [0, 1]. We consider two cases: Case I and Case II. In Case I, each client randomly samples its α value from the interval [0, 1] before the training process begins, and then α remains fixed for that client throughout the training process. In Case II, each client randomly samples its α value from the interval [0, 1] in each training round, i.e., α changes during training for each client. Note that in Case I and Case II, all clients still use the same aggregation rule. The results for Case I are shown in Table 11 in Appendix, the results for Case II are shown in Table 12 in Appendix. Comparing Table 2a and Table 11, we observe that having different clients use different values of α cannot reduce the impact of poisoning attacks. Moreover, when all clients are benign, the Max.TERs of DFL methods including our BALANCE are large compared to the scenario when clients using the same α. Our proposed BALANCE achieves similar Max.TERs under different attacks compared to the FedAvg method without any attacks. However, existing defenses are still vulnerable to poisoning attacks.

We then study the scenario where different clients use different aggregation rules to combine the received neighboring clients' models. We also investigate two cases, namely Case III and Case IV. In both cases, we randomly assign one existing Byzantine-robust aggregation rule from set $\mathcal{H}$ ={Krum, Trim-mean, Median, FLTrust, UBAR, SCCLIP} to each client. Note that the set $\mathcal{H}$ excludes FedAvg, LEARN, and our proposed method. This is because FedAvg is not robust; LEARN method requires exchanging both local model updates and local models between clients, while other methods only need to exchange local models; and our proposed method is already robust and does not require being used with other aggregation rules. Specifically, in Case III, each client $i \in \mathcal{V}$ randomly selects one aggregation rule from the set $\mathcal{H}$ before the training process. In Case IV, in each training round, client i randomly selects one robust Byzantine-robust aggregation rule from the set $\mathcal{H}$. The results are shown in Table 13 in Appendix. Compare Table 2a and Table 13, we observe that when different aggregation rules are used by clients, they are more susceptible to poisoning attacks compared to the scenario where all clients use the same aggregation rule.

Table 3: Results of different DFL methods with time-varying communication graph.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.15	0.90	0.90	0.91	0.90	0.90 / 1.00	0.90
Krum	0.13	0.15	0.90	0.90	0.15	0.14	0.90 / 1.00	0.90
Trim-mean	0.27	0.27	0.90	0.90	0.91	0.91	0.90 / 1.00	0.90
Median	0.24	0.28	0.90	0.90	0.91	0.91	0.90 / 1.00	0.90
FLTrust	0.11	0.16	0.90	0.89	0.89	0.89	0.89 / 0.09	0.89
UBAR	0.15	0.17	0.90	0.90	0.15	0.19	0.15 / 0.01	0.90
LEARN	0.19	0.19	0.38	0.90	0.91	0.91	0.90 / 1.00	0.90
SCCLIP	0.11	0.16	0.34	0.15	0.91	0.91	0.23 / 0.02	0.64
BALANCE	0.11	0.12	0.12	0.12	0.11	0.12	0.11 / 0.01	0.12

Time-varying communication graph: By default, we consider a static communication graph $\mathcal{G}$, i.e., once established, $\mathcal{G}$ is fixed. Here we consider a practical setting, where each client has a chance of disconnecting from the protocol, such as Internet issues. When a client disconnects from the protocol during a specific round, it is unable to exchange information with its neighboring clients. However, a disconnected client can continue training its model locally, and it may reconnect to the protocol in the subsequent round. We consider the default parameter settings, where there are 4 out of 20 clients are malicious, MNIST dataset and regular-(20, 10) graph are used (client may disconnect from the protocol based on the regular-(20, 10) graph). However, each client has 20% possibility of disconnecting from the protocol. The results are shown in Table 3. We observe that existing defenses are more vulnerable to poisoning attacks, while BALANCE could still defend against various attacks.

Impact of the total number of clients: Fig. 10 in Appendix shows the results of different methods on MNIST dataset when the total number of clients is changed, while keeping the fraction of malicious clients fixed at 20%. Note that we use regular-(10, 5), regular-(20, 10), regular-(30, 15), regular-(40, 20) and regular-(50, 25) communication graphs when we have 10, 20, 30, 40 and 50 clients in total, respectively. The topologies of regular-(20, 10) and regular-(30, 15) graphs are shown in Figs. 9a-9b. The topologies of regular-(10, 5), regular-(40, 20) and regular-(50, 25) graphs are illustrated in Figs. 9c-9e. We observe from Fig. 10 that our proposed DFL approach is resilient to poisoning attacks for all the considered total number of clients ranging from 10 to 50.

Different initial models or communication graphs: We assume that all clients use the same initial local model by default. We also explore the setting where different clients use different initial local models, other parameters are set to their default settings. The results are shown in Table 14 in Appendix. We observe that our proposed method is robust against poisoning attacks and outperforms baselines, even when clients use different initial models.

We also investigate other types of communication graphs, including complete graph, Erdős–Rényi graph, small-world graph and ring graph, the topologies of these four graphs are illustrated in Figs. 9f-9i in Appendix. Note that for a complete graph, each client is connected to the remaining clients; for a ring graph, clients form a ring, and each client only has two neighbors. In all four communication graphs, there are 20 clients in total, and 4 clients are malicious. We maintain default settings for other parameters. The results for four graphs are shown in Table 15 in Appendix. We

Table 4: Results of different DFL methods, where each client only has three classes of training data.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.18	0.90	0.91	0.91	0.90	0.90 / 1.00	0.91
Krum	0.46	0.49	0.91	0.49	0.57	0.52	0.47 / 0.16	0.46
Trim-mean	0.17	0.62	0.27	0.24	0.58	0.83	0.79 / 0.23	0.81
Median	0.56	0.89	0.56	0.88	0.64	0.70	0.56 / 0.04	0.58
FLTrust	0.10	0.18	0.91	0.22	0.18	0.90	0.20 / 0.27	0.14
UBAR	0.42	0.42	0.91	0.52	0.52	0.57	0.62 / 0.32	0.54
LEARN	0.10	0.15	0.10	0.15	0.20	0.61	0.16 / 0.09	0.58
SCCLIP	0.10	0.18	0.23	0.21	0.90	0.91	0.33 / 0.08	0.89
BALANCE	0.10	0.14	0.13	0.14	0.14	0.14	0.11 / 0.02	0.14

observe that our BALANCE can also achieve Byzantine-robustness when other types of communication graphs are used.

Impact of fraction of edges between malicious and benign clients: Malicious clients in a DFL system attempt to manipulate the system by sharing harmful information, such as carefully crafted local models, with their neighboring clients. The attack performance is generally influenced by the *Fraction of Edges between Malicious and Benign clients (FEMB)*. In our paper, FEMB is calculated as the ratio of the number of edges between malicious and benign clients to the total number of edges in the communication graph. In our experiments, we generate three random graphs to study the impact of FEMB on the attack performance. For all three random graphs, there are 20 nodes in total, each node represents one client, and 4 out of 20 clients are malicious. The FEMBs for the three graphs are 0.16, 0.22, and 0.32, respectively. The topologies of three graphs are shown in Figs. 9j-9l. All other parameters are kept at their default values. The experimental results are shown in Table 16 in Appendix. We observe that the attack performance generally increases with an increase in FEMB.

7 DISCUSSION AND LIMITATIONS

More extreme Non-IID distribution: In our default Non-IID setting, a client's primary training data come from just one class, with few examples from other classes. This section explores a more extreme Non-IID scenario as described in [35]. Here, the distribution of training data among clients is based solely on labels, and each client having training data from only three classes. For instance, Client 1 possesses training data only for labels 0-2, while Client 2 exclusively holds data for labels 3-5.

The results of various DFL methods under various attacks with this more extreme Non-IID scenario are shown in Table 4. When comparing Table 2a with Table 4, it becomes apparent that in this Non-IID context, the Max.TER of current DFL methods is significantly high, even in non-adversarial setting. For instance, with the UBAR aggregation rule and all clients being benign, the Max.TER reaches 0.42. In adversarial setting, our suggested BALANCE continues to effectively counter all the poisoning attacks considered, achieving the largest Max.TER of just 0.14. Nonetheless, existing DFL approaches show increased susceptibility to poisoning attacks. To illustrate, the Max.TER of Median is 0.89 under LF attack.

More adaptive and decentralized attacks: In our prior experiments, we show that our proposed BALANCE is robust against data poisoning attacks (such as LF and Feature attacks) as well as sophisticated adaptive attacks. For these data poisoning attacks, attackers corrupt the local training data on malicious clients. While the models trained on this poisoned data might appear benign, our experiments reveal that such attacks are ineffectual against our BALANCE due to their limited impact. In the case of the adaptive attacks we considered, the attacker, aware of our aggregation rule, introduces subtle, strategic perturbation to the benign local models in order to circumvent our defenses. Nonetheless, our experimental results indicate that as attackers adapt their strategies to evade our defense mechanism, the efficacy of their attacks diminishes.

Table 5: Results of different DFL methods under LIE and Dissensus attacks.

Method	No	LIE	Dissensus
FedAvg	0.10	0.13	0.91
Krum	0.10	0.10	0.15
Trim-mean	0.11	0.16	0.86
Median	0.14	0.18	0.87
FLTrust	0.10	0.10	0.12
UBAR	0.14	0.14	0.23
LEARN	0.10	0.10	0.69
SCCLIP	0.10	0.10	0.91
BALANCE	0.10	0.10	0.10

Table 6: Results of different variants of BALANCE.

Variant	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
Variant I	0.10	0.11	0.12	0.11	0.11	0.11	0.11 / 0.01	0.14
Variant II	0.10	0.10	0.13	0.16	0.11	0.18	0.11 / 0.01	0.16
BALANCE	0.10	0.10	0.11	0.10	0.10	0.11	0.11 / 0.01	0.11

In this section, we first explore a different type of adaptive attack known as the "a little is enough" (LIE) attack [3]. The LIE attack represents a general attack model where the attacker is not required to be aware of the aggregation rule employed by other clients. In executing a LIE attack, the attacker calculates the variance among benign models, and then introduces minimal changes to these models, intended to circumvent the aggregation rule. Our current experiments demonstrate that extending server-based attacks to the DFL setting can be effective against prevailing DFL methods. Additionally, we examine the Dissensus attack [22], a newly devised attack model specifically tailored for DFL systems. In the Dissensus attack, the attacker designs malicious local models with the intention of disrupting consensus among benign clients.

Table 5 presents the results of various DFL methods under LIE and Dissensus attacks. It is evident from the results that the LIE attack does not significantly compromise existing DFL methods. This could be attributed to the fact that the LIE attack is a more generic attack model and is not specifically tailored for a fully decentralized environment. In contrast, the Dissensus attack shows a considerable ability to disrupt DFL methods, particularly in the case of Trim-mean and Median. For instance, under the Dissensus attack, the Max.TER for the Median method escalates to 0.87.

Different variants of BALANCE: In this part, we consider two variants of our proposed BALANCE.

- **Variant I:** In this variant, client i accepts $\boldsymbol{w}_j^{t+\frac{1}{2}}$ when the condition $\|\boldsymbol{w}_i^{t+\frac{1}{2}} - \boldsymbol{w}_j^{t+\frac{1}{2}}\| \leq \gamma \|\boldsymbol{w}_i^{t+\frac{1}{2}}\|$ holds true.

- **Variant II:** Client i calculates $q_j = \frac{\|\boldsymbol{w}_i^{t+\frac{1}{2}} - \boldsymbol{w}_j^{t+\frac{1}{2}}\|}{\|\boldsymbol{w}_i^{t+\frac{1}{2}}\|}$ for each neighbor j in its neighbor set $\mathcal{N}_i$. After computing the median of these values from the $|\mathcal{N}_i|$ neighbors, denoted as q_{med} where $q_{\text{med}} = \text{med}\{q_1, ..., q_{|\mathcal{N}_i|}\}$, client i will accept $\boldsymbol{w}_j^{t+\frac{1}{2}}$ if $q_j \leq \min\{q_{\text{med}}, \gamma\}$. The γ is set to safeguard against the possibility of most neighbors being malicious. The key idea of Variant II is that client i will

accept client j's local model if it is close to its own, based on a comparison with the median of deviations from all neighbors.

Table 6 compares our BALANCE with two variants. Variant II underperforms due to its tendency to incorrectly reject many benign local models. In contrast, Variant I shows performance comparable to our BALANCE. Note that the distance between $\boldsymbol{w}_i^{t+\frac{1}{2}}$ and $\boldsymbol{w}_j^{t+\frac{1}{2}}$ will become smaller, as the DFL system approaches convergence, leading Variant I to inadvertently accept some malicious models in later training stages. Nevertheless, attacking the DFL system becomes challenging as the model nears convergence. Additional experiments support this claim, showing that the Max.TER of Median under Trim attack is 0.55 when attacks occur only in the first half of training rounds (1-1,000 rounds out of 2,000 total), while it is 0.38 when attacks only occur in the second half. When attacks happen in all rounds, the Max.TER is 0.63. In our BALANCE, training the model for a sufficient number of rounds (a large T) can significantly reduce the value of $\gamma \cdot \exp(-\kappa \cdot \lambda(t))$. By selecting a smaller κ, we can decelerate the decline of the exponential function.

8 CONCLUSION AND FUTURE WORK

In this work, we proposed a novel method called BALANCE to defend against poisoning attacks in DFL. In our proposed method, each client uses its local model as a reference point to check whether the received neighboring client's local model is malicious or benign. We established the convergence performance of our method under poisoning attacks in both strongly convex and non-convex settings, and the convergence rate of our BALANCE matches those of the state-of-the-art counterparts in Byzantine-free settings. Extensive experiments across various settings demonstrated the efficacy of our proposed method. Our future work includes designing an optimized strategy to dynamically select aggregation rules and parameter α for different clients to enhance the robustness of existing DFL methods.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their comments. This work was supported by NSF grants CAREER CNS-2110259, CNS-2112471, CNS-2312138, SaTC-2350075, No. 2131859, 2125977, 2112562, 1937786, 1937787, and ARO grant No. W911NF2110182.

REFERENCES

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra Perez, and Jorge Luis Reyes Ortiz. 2013. A public domain dataset for human activity recognition using smartphones. In *ESANN*.
[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *AISTATS*.
[3] Gilad Baruch, Moran Baruch, and Yoav Goldberg. 2019. A little is enough: Circumventing defenses for distributed learning. In *NeurIPS*.
[4] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. 2022. Decentralized Federated Learning: Fundamentals, State-of-the-art, Frameworks, Trends, and Challenges. In *arXiv preprint arXiv:2211.08413*.
[5] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. In *ICML*.
[6] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NeurIPS*.
[7] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).
[8] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. 2021. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *NDSS*.
[9] Xiaoyu Cao and Neil Zhenqiang Gong. 2022. Mpaf: Model poisoning attacks to federated learning based on fake clients. In *CVPR Workshops*.
[10] Xiaoyu Cao, Jinyuan Jia, Zaixi Zhang, and Neil Zhenqiang Gong. 2023. Fedrecover: Recovering from poisoning attacks in federated learning using historical information. In *IEEE Symposium on Security and Privacy*.
[11] Xiaoyu Cao, Zaixi Zhang, Jinyuan Jia, and Neil Zhenqiang Gong. 2022. Flcert: Provably secure federated learning against poisoning attacks. *IEEE Transactions on Information Forensics and Security*.
[12] Tianyue Chu, Alvaro Garcia-Recuero, Costas Iordanou, Georgios Smaragdakis, and Nikolaos Laoutaris. 2023. Securing Federated Sensitive Topic Classification against Poisoning Attacks. In *NDSS*.
[13] Rong Dai, Li Shen, Fengxiang He, Xinmei Tian, and Dacheng Tao. 2022. Dispfl: Towards communication-efficient personalized federated learning via decentralized sparse training. In *ICML*.
[14] El Mahdi El-Mhamdi, Sadegh Farhadkhani, Rachid Guerraoui, Arsany Guirguis, Lê-Nguyên Hoang, and Sébastien Rouault. 2021. Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and nonconvex learning). In *NeurIPS*.
[15] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to Byzantine-robust federated learning. In *USENIX Security Symposium*.
[16] Minghong Fang, Jia Liu, Neil Zhenqiang Gong, and Elizabeth S Bentley. 2022. Afluard: Byzantine-robust asynchronous federated learning. In *ACSAC*.
[17] Liam Fowl, Jonas Geiping, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. 2022. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *ICLR*.
[18] Shuhao Fu, Chulin Xie, Bo Li, and Qifeng Chen. 2019. Attack-resistant federated learning with residual-based reweighting. *arXiv preprint arXiv:1912.11464* (2019).
[19] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2020. The Limitations of Federated Learning in Sybil Settings. In *RAID*.
[20] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).
[21] Shangwei Guo, Tianwei Zhang, Han Yu, Xiaofei Xie, Lei Ma, Tao Xiang, and Yang Liu. 2021. Byzantine-resilient decentralized stochastic gradient descent. In *IEEE Transactions on Circuits and Systems for Video Technology*.
[22] Lie He, Sai Praneeth Karimireddy, and Martin Jaggi. 2023. Byzantine-robust decentralized learning via self-centered clipping. *arXiv preprint arXiv:2202.01545* (2023).
[23] Shivam Kalra, Junfeng Wen, Jesse C Cresswell, Maksims Volkovs, and HR Tizhoosh. 2023. Decentralized federated learning through proxy model sharing. In *Nature Communications*.
[24] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. 2022. Byzantine-robust learning on heterogeneous datasets via bucketing. In *ICLR*.
[25] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. In *NeurIPS Workshop on Private Multi-Party Machine Learning*.
[26] Lingjing Kong, Tao Lin, Anastasia Koloskova, Martin Jaggi, and Sebastian Stich. 2021. Consensus control for decentralized deep learning. In *ICML*.
[27] Kavita Kumari, Phillip Rieger, Hossein Fereidooni, Murtuza Jadliwala, and Ahmad-Reza Sadeghi. 2023. BayBFed: Bayesian Backdoor Defense for Federated Learning. In *IEEE Symposium on Security and Privacy*.
[28] Anusha Lalitha, Shubhanshu Shekhar, Tara Javidi, and Farinaz Koushanfar. 2018. Fully decentralized federated learning. In *Third workshop on bayesian deep learning (NeurIPS)*.
[29] Yann LeCun, Corinna Cortes, and CJ Burges. 1998. MNIST handwritten digit database. *Available: http://yann. lecun. com/exdb/mnist* (1998).
[30] Henger Li, Xiaolin Sun, and Zizhan Zheng. 2022. Learning to attack federated learning: A model-based reinforcement learning attack framework. In *NeurIPS*.
[31] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. 2017. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *NeurIPS*.
[32] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *ICCV*.
[33] Zhuqing Liu, Xin Zhang, Prashant Khanduri, Songtao Lu, and Jia Liu. 2023. Prometheus: taming sample and communication complexities in constrained decentralized stochastic bilevel learning. In *ICML*.
[34] Songtao Lu, Yawen Zhang, and Yunlong Wang. 2020. Decentralized Federated Learning for Electronic Health Records. In *CISS*.
[35] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.
[36] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. 2018. The Hidden Vulnerability of Distributed Learning in Byzantium. In *ICML*.
[37] Hamid Mozaffari, Virat Shejwalkar, and Amir Houmansadr. 2023. Every Vote Counts: Ranking-Based Training of Federated Learning to Resist Poisoning Attacks. In *USENIX Security Symposium*.

[38] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *AISec*.

[39] Luis Muñoz-González, Kenneth T Co, and Emil C Lupu. 2019. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125* (2019).

[40] Yurii Nesterov et al. 2018. *Lectures on convex optimization*. Vol. 137. Springer.

[41] Anh Nguyen, Tuong Do, Minh Tran, Binh X Nguyen, Chien Duong, Tu Phan, Erman Tjiputra, and Quang D Tran. 2022. Deep federated learning for autonomous driving. In *IEEE Intelligent Vehicles Symposium*.

[42] TV Nguyen, MA Dakka, SM Diakiw, MD VerMilyea, M Perugini, JMM Hall, and D Perugini. [n.d.]. A novel decentralized federated learning approach to train on globally distributed, poor quality, and protected private medical data. In *Scientific Reports*.

[43] Xudong Pan, Mi Zhang, Duocai Wu, Qifan Xiao, Shouling Ji, and Min Yang. 2020. Justinian's gaavernor: Robust distributed learning with gradient aggregation agent. In *USENIX Security Symposium*.

[44] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. 2022. Eluding secure aggregation in federated learning via model inconsistency. In *CCS*.

[45] Dario Pasquini, Mathilde Raynal, and Carmela Troncoso. 2023. On the (In)security of Peer-to-Peer Decentralized Machine Learning. In *IEEE Symposium on Security and Privacy*.

[46] Boris Teodorovich Polyak. 1963. Gradient methods for the minimisation of functionals. In *USSR Computational Mathematics and Mathematical Physics*.

[47] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. 2020. The future of digital health with federated learning. In *NPJ digital medicine*.

[48] Virat Shejwalkar and Amir Houmansadr. 2021. Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning. In *NDSS*.

[49] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. 2022. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *IEEE Symposium on Security and Privacy*.

[50] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In *ESORICS*.

[51] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:cs.LG/cs.LG/1708.07747

[52] Chulin Xie, Minghao Chen, Pin-Yu Chen, and Bo Li. 2021. Crfl: Certifiably robust federated learning against backdoor attacks. In *ICML*.

[53] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. 2020. Dba: Distributed backdoor attacks against federated learning. In *ICLR*.

[54] Yichang Xu, Ming Yin, Minghong Fang, and Neil Zhenqiang Gong. 2024. Robust Federated Learning Mitigates Client-side Training Data Distribution Inference Attacks. In *The Web Conference*.

[55] Gokberk Yar, Cristina Nita-Rotaru, and Alina Oprea. 2023. Backdoor Attacks in Peer-to-Peer Federated Learning. *arXiv preprint arXiv:2301.09732* (2023).

[56] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. 2018. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *ICML*.

[57] Ming Yin, Yichang Xu, Minghong Fang, and Neil Zhenqiang Gong. 2024. Poisoning Federated Recommender Systems with Fake Users. In *The Web Conference*.

[58] Xin Zhang, Minghong Fang, Zhuqing Liu, Haibo Yang, Jia Liu, and Zhengyuan Zhu. 2022. Net-fleet: Achieving linear convergence speedup for fully decentralized federated learning with heterogeneous data. In *MobiHoc*.

[59] Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2022. FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *KDD*.

[60] Zifan Zhang, Minghong Fang, Jiayuan Huang, and Yuchen Liu. 2024. Poisoning Attacks on Federated Learning-based Wireless Traffic Prediction. In *IFIP/IEEE Networking*.

A PROOF OF THEOREM 1

We denote by $\boldsymbol{g}(\boldsymbol{w}_i^t)$ the stochastic gradient that client i computed based on the model $\boldsymbol{w}_i^t$, then we have that:

$$\boldsymbol{w}_i^{t+\frac{1}{2}} = \boldsymbol{w}_i^t - \eta \boldsymbol{g}(\boldsymbol{w}_i^t), \tag{5}$$

where $\eta > 0$ is the learning rate.

In the following proof, we ignore the superscript t in $\mathcal{S}_i^t$ for simplicity. Thus we have that:

$$\begin{aligned}
&\boldsymbol{w}_i^{t+1} - \boldsymbol{w}_i^t \\
&\overset{(a)}{=} \alpha \boldsymbol{w}_i^{t+\frac{1}{2}} + (1-\alpha)\mathrm{AGG}\{\boldsymbol{w}_j^{t+\frac{1}{2}}, j \in \mathcal{N}_i\} - \boldsymbol{w}_i^t \\
&\overset{(b)}{=} \alpha \boldsymbol{w}_i^{t+\frac{1}{2}} + (1-\alpha)\frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}} + \boldsymbol{w}_i^{t+\frac{1}{2}}) - \boldsymbol{w}_i^t \\
&= [\alpha + (1-\alpha)]\boldsymbol{w}_i^{t+\frac{1}{2}} + \frac{1-\alpha}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}}) - \boldsymbol{w}_i^t \\
&= \boldsymbol{w}_i^{t+\frac{1}{2}} + \frac{1-\alpha}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}}) - \boldsymbol{w}_i^t \\
&\overset{(c)}{=} [\boldsymbol{w}_i^t - \eta \boldsymbol{g}(\boldsymbol{w}_i^t) - \boldsymbol{w}_i^t] + \frac{1-\alpha}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}}) \\
&= -\eta \boldsymbol{g}(\boldsymbol{w}_i^t) + \frac{1-\alpha}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}}),
\end{aligned} \tag{6}$$

where (a) is because of Eq. (2); (b) is due to Eq. (4); (c) is due to Eq. (5).

According to Assumption 2, when the loss function is L-smooth, one has that:

$$\begin{aligned}
F(\boldsymbol{w}_i^{t+1}) \le F(\boldsymbol{w}_i^t) + \left\langle \nabla F(\boldsymbol{w}_i^t), \boldsymbol{w}_i^{t+1} - \boldsymbol{w}_i^t \right\rangle \\
+ \frac{L}{2}\left\| \boldsymbol{w}_i^{t+1} - \boldsymbol{w}_i^t \right\|^2.
\end{aligned} \tag{7}$$

Combining Eq. (6) and Eq. (7), we obtain:

$$\begin{aligned}
&F(\boldsymbol{w}_i^{t+1}) \\
&\le F(\boldsymbol{w}_i^t) + \langle \nabla F(\boldsymbol{w}_i^t), -\eta \boldsymbol{g}(\boldsymbol{w}_i^t) + \frac{1-\alpha}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\rangle \\
&\quad + \frac{L}{2}\| - \eta \boldsymbol{g}(\boldsymbol{w}_i^t) + \frac{1-\alpha}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\|^2 \\
&\overset{(a)}{\le} F(\boldsymbol{w}_i^t) - \eta\langle \nabla F(\boldsymbol{w}_i^t), \boldsymbol{g}(\boldsymbol{w}_i^t)\rangle \\
&\quad + \langle \nabla F(\boldsymbol{w}_i^t), \frac{1-\alpha}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\rangle + L\eta^2\|\boldsymbol{g}(\boldsymbol{w}_i^t)\|^2 \\
&\quad + L\|\frac{1-\alpha}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\|^2,
\end{aligned} \tag{8}$$

where (a) is because of Lemma 1 in Appendix C.

Taking expectation on both sides of Eq. (8), one obtains:

$$\begin{aligned}
&\mathbb{E}[F(\boldsymbol{w}_i^{t+1})] \\
&\overset{(a)}{\le} \mathbb{E}[F(\boldsymbol{w}_i^t) - \eta\|\nabla F(\boldsymbol{w}_i^t)\|^2 \\
&\quad + (1-\alpha)\langle \nabla F(\boldsymbol{w}_i^t), \frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\rangle \\
&\quad + L\eta^2\|\boldsymbol{g}(\boldsymbol{w}_i^t) - \nabla F(\boldsymbol{w}_i^t) + \nabla F(\boldsymbol{w}_i^t)\|^2 \\
&\quad + L(1-\alpha)^2\|\frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\|^2] \\
&\overset{(b)}{\le} \mathbb{E}[F(\boldsymbol{w}_i^t) - \eta\|\nabla F(\boldsymbol{w}_i^t)\|^2 + 2L\eta^2\|\nabla F(\boldsymbol{w}_i^t)\|^2 \\
&\quad + (1-\alpha)\langle \nabla F(\boldsymbol{w}_i^t), \frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\rangle \\
&\quad + 2L\eta^2\delta^2 + L(1-\alpha)^2\|\frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\|^2]
\end{aligned}$$

Table 7: The CNN architecture.

Layer	Size
Input	$28 \times 28 \times 1$
Convolution + ReLU	$3 \times 3 \times 30$
Max Pooling	2×2
Convolution + ReLU	$3 \times 3 \times 50$
Max Pooling	2×2
Fully Connected + ReLU	100
Softmax	10

$$
\begin{aligned}
&= \mathbb{E}[F(\boldsymbol{w}_i^t) - (\eta - 2L\eta^2)\|\nabla F(\boldsymbol{w}_i^t)\|^2 + 2L\eta^2\delta^2 \\
&\quad + (1-\alpha)\langle \nabla F(\boldsymbol{w}_i^t), \frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\rangle \\
&\quad + L(1-\alpha)^2\|\frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\|^2] \\
&\overset{(c)}{\leq} \mathbb{E}[F(\boldsymbol{w}_i^t) - \eta(1-2L\eta)\|\nabla F(\boldsymbol{w}_i^t)\|^2 + 2L\eta_t^2\delta^2 \\
&\quad + (1-\alpha)\|\nabla F(\boldsymbol{w}_i^t)\| \cdot \|\frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\| \\
&\quad + L(1-\alpha)^2\|\frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\|^2], \qquad (9)
\end{aligned}
$$

where (a) is because of Assumption 3 that $\mathbb{E}[\boldsymbol{g}(\boldsymbol{w}_i^t)] = \nabla F(\boldsymbol{w}_i^t)$; (b) is due to Lemma 1 in Appendix C, and Assumption 3 that $\mathbb{E}[\|\boldsymbol{g}(\boldsymbol{w}_i^t) - \nabla F(\boldsymbol{w}_i^t)\|]^2 \leq \delta^2$; (c) is because of Cauchy-Schwarz inequality.

Next, we bound the term $\|\frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\|$:

$$
\begin{aligned}
&\|\frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\| = \|\frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\| \\
&\leq \frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}\|\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}}\| \overset{(a)}{\leq} \frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}\gamma\|\boldsymbol{w}_i^{t+\frac{1}{2}}\| \overset{(b)}{\leq} \gamma\psi,
\end{aligned}
$$

where (a) is due to Eq. (3) and $\gamma \cdot \exp(-\kappa \cdot \lambda(t)) \leq \gamma$; (b) is because of Assumption 4 that $\|\boldsymbol{w}_i^t\| \leq \psi$.

Due to Assumption 4, then we have:

$$
\|\nabla F(\boldsymbol{w}_i^t)\| \cdot \|\frac{1}{|\mathcal{S}_i|}\sum_{j\in\mathcal{S}_i}(\boldsymbol{w}_j^{t+\frac{1}{2}} - \boldsymbol{w}_i^{t+\frac{1}{2}})\| \leq \gamma\psi\rho. \qquad (10)
$$

If γ satisfies $\gamma \leq \frac{\rho}{L\psi(1-\alpha)}$, then we have $L\gamma^2\psi^2(1-\alpha)^2 \leq \gamma\rho\psi(1-\alpha)$, and we further have:

$$
\begin{aligned}
\mathbb{E}[F(\boldsymbol{w}_i^{t+1})] &\leq \mathbb{E}[F(\boldsymbol{w}_i^t) - \eta(1-2L\eta)\|\nabla F(\boldsymbol{w}_i^t)\|^2 \\
&\quad + 2L\eta^2\delta^2 + 2\gamma\rho\psi(1-\alpha)]. \qquad (11)
\end{aligned}
$$

If the learning rate η satisfies $\eta \leq \frac{1}{4L}$, one has that:

$$
\begin{aligned}
\mathbb{E}[F(\boldsymbol{w}_i^{t+1})] &\leq \mathbb{E}[F(\boldsymbol{w}_i^t) - \frac{\eta_t}{2}\|\nabla F(\boldsymbol{w}_i^t)\|^2 + 2L\eta^2\delta^2 \\
&\quad + 2\gamma\rho\psi(1-\alpha)]. \qquad (12)
\end{aligned}
$$

By Assumption 1, since $F(\cdot)$ is μ-strongly convex, then one has the following Polyak-Łojasiewicz (PL) inequality [46] $\|\nabla F(\boldsymbol{w}_i^t)\|^2 \geq 2\mu(F(\boldsymbol{w}_i^t) - F(\boldsymbol{w}^*))$. Therefore, we further have that:

$$
\begin{aligned}
\mathbb{E}[F(\boldsymbol{w}_i^{t+1})] &\leq \mathbb{E}[F(\boldsymbol{w}_i^t) - \mu\eta(F(\boldsymbol{w}_i^t) - F(\boldsymbol{w}^*)) \\
&\quad + 2L\eta^2\delta^2 + 2\gamma\rho\psi(1-\alpha)]. \qquad (13)
\end{aligned}
$$

Subtracting $F(\boldsymbol{w}^*)$ on both sides, we get:

$$
\begin{aligned}
\mathbb{E}[F(\boldsymbol{w}_i^{t+1}) - F(\boldsymbol{w}^*)] &\leq (1-\mu\eta)\,\mathbb{E}[F(\boldsymbol{w}_i^t) - F(\boldsymbol{w}^*)] \\
&\quad + 2L\eta^2\delta^2 + 2\gamma\rho\psi(1-\alpha)]. \qquad (14)
\end{aligned}
$$

By choosing $\eta \leq \frac{1}{\mu}$, then we can guarantee that $1 - \mu\eta \geq 0$. Telescoping over $t = 0, 1, ..., T-1$, one has that:

$$
\begin{aligned}
&\mathbb{E}[F(\boldsymbol{w}_i^{t+1}) - F(\boldsymbol{w}^*)] \\
&\leq (1-\mu\eta)^T[F(\boldsymbol{w}_i^0) - F(\boldsymbol{w}^*)] + \sum_{t=0}^{T-1}(1-\mu\eta)^T 2L\eta^2\delta^2 \\
&\quad + \sum_{t=0}^{T-1}(1-\mu\eta)^T 2\gamma\rho\psi(1-\alpha) \\
&= (1-\mu\eta)^T[F(\boldsymbol{w}_i^0) - F(\boldsymbol{w}^*)] + \frac{2L\eta\delta^2}{\mu} + \frac{2\gamma\rho\psi(1-\alpha)}{\mu\eta}.
\end{aligned}
$$

B PROOF OF THEOREM 2

According to Eq. (12), we have the following:

$$
\begin{aligned}
\mathbb{E}[F(\boldsymbol{w}_i^{t+1})] &\leq \mathbb{E}[F(\boldsymbol{w}_i^t) - \frac{\eta_t}{2}\|\nabla F(\boldsymbol{w}_i^t)\|^2 + 2L\eta^2\delta^2 \\
&\quad + 2\gamma\rho\psi(1-\alpha)]. \qquad (15)
\end{aligned}
$$

Rearranging the term, one obtains that:

$$
\begin{aligned}
\frac{\eta_t}{2}\,\mathbb{E}[\|\nabla F(\boldsymbol{w}_i^t)\|^2] &\leq \mathbb{E}[F(\boldsymbol{w}_i^t) - F(\boldsymbol{w}_i^{t+1})] + 2L\eta^2\delta^2 \\
&\quad + 2\gamma\rho\psi(1-\alpha)]. \qquad (16)
\end{aligned}
$$

By telescoping over $t = 0, 1, ..., T-1$ and taking an average over T, then we get:

$$
\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\boldsymbol{w}_i^t)\|^2] &\leq \frac{2[F(\boldsymbol{w}_i^0) - F(\boldsymbol{w}^*)]}{\eta T} + 4L\eta\delta^2 \\
&\quad + \frac{4\gamma\rho\psi(1-\alpha)}{\eta}. \qquad (17)
\end{aligned}
$$

C USEFUL TECHNICAL LEMMA

Lemma 1. *Given a set of vectors $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ for all $i \in \{0, 1, \cdots, n-1\}$, we have the following:*

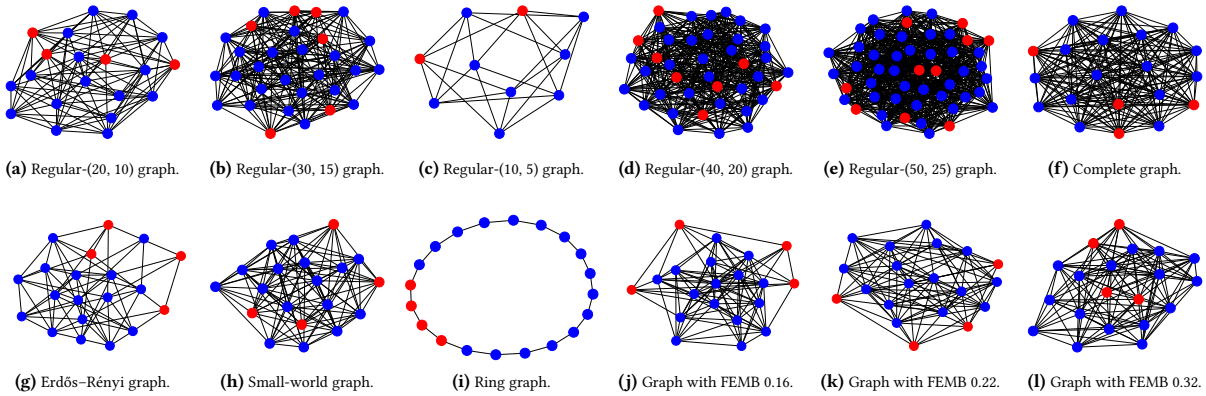
$$
\|\sum_{i=0}^{n-1}\boldsymbol{x}_i\|^2 \leq n\sum_{i=0}^{n-1}\|\boldsymbol{x}_i\|^2.
$$

D DATASETS AND POISONING ATTACKS

D.1 Datasets

Synthetic: To create synthetic data, we model the dependent variable as $y = \langle \boldsymbol{x}, \boldsymbol{w}^* \rangle + \epsilon$, with $\boldsymbol{x}$ as a feature vector, $\boldsymbol{w}^*$ as the true parameter (100-dimensional), and ϵ as noise. We generate $\boldsymbol{x}$ and ϵ from a standard normal distribution $N(0,1)$, and $\boldsymbol{w}^*$ from $N(0,25)$. Our dataset comprises 10,000 instances, split into 8,000 for training and 2,000 for testing.

MNIST [29]: This dataset has 10 classes, which contains 60,000 images for training and 10,000 images for testing.



(a) Regular-(20, 10) graph. **(b)** Regular-(30, 15) graph. **(c)** Regular-(10, 5) graph. **(d)** Regular-(40, 20) graph. **(e)** Regular-(50, 25) graph. **(f)** Complete graph.

(g) Erdős–Rényi graph. **(h)** Small-world graph. **(i)** Ring graph. **(j)** Graph with FEMB 0.16. **(k)** Graph with FEMB 0.22. **(l)** Graph with FEMB 0.32.

Figure 9: Different communication graphs.

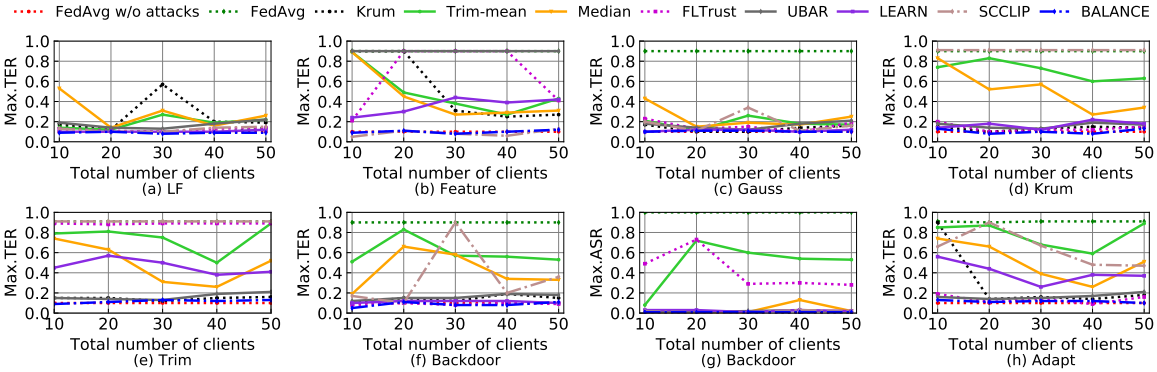


Figure 10: Impact of total number of clients.

Fashion-MNIST [51]: Each image in Fashion-MNIST belongs to one of the 10 categories. The training data contains 60,000 images, and the testing set contains 10,000 images.

Human Activity Recognition (HAR) [1]: The HAR dataset aims to classify six human activities, collected from 30 smartphone users, totaling 10,299 instances with 561 features each. Follow [8], we randomly use 75% of each user's data for training and 25% for testing.

Large-scale CelebFaces Attributes (CelebA) [32]: CelebA is a large-scale image dataset that identifies celebrity face attributes. This data contains 200,288 images, which includes 177,480 images for training and 22,808 images for testing. Each image has 40 annotations of binary attributes. Following [7], we consider the binary classification task for the CelebA dataset, which aims to predict whether the person in the image is smiling or not.

D.2 Poisoning Attack Schemes

Label flipping (LF) attack [50]: In the synthetic data, the attacker modifies malicious clients' local training data by adding a bias of 5 to the dependent variable y. For MNIST, Fashion-MNIST, and HAR datasets, training labels on malicious clients are changed from class 3 to class 5, following [50]. In CelebA, labels are reversed on malicious clients, switching 0 to 1 and vice versa.

Feature attack: The attacker modifies the features of local training examples on malicious clients. Each feature of such examples is replaced with a value drawn from a Gaussian distribution with a mean of 0 and a variance of 1,000.

Gaussian (Gauss) attack [6]: Malicious clients send Gaussian vectors, randomly drawn from a normal distribution with a mean of 0 and a variance of 200, to their neighbors.

Krum attack [15]: The attacker carefully crafts the local models on malicious clients in a way that causes the Krum rule to output the model chosen by the attacker.

Trim attack [15]: The attacker in Trim attack manipulates the local models on malicious clients such that the aggregated local model after attack deviates significantly from the before-attack aggregated one.

Backdoor attack [2, 20]: Malicious clients replicate their training data, adding a backdoor trigger to each copy and assigning them a target label chosen for the attack. They train their local models using this augmented data and share the scaled models with neighbors. The scaling factor equals the total number of clients. We use the triggers suggested in [8] for the MNIST, Fashion-MNIST, and HAR datasets. For the CelebA dataset, we set the first binary feature to 1.

Adaptive (Adapt) attack [48]: We consider the adaptive attack proposed in [48]. We consider the worst-case attack setting, where

the attacker is aware of the aggregation rule used by each client, i.e., BALANCE in our paper, and the local models of benign clients.

D.3 Consensus Error

To assess disagreement among benign clients during poisoning attacks, we use the consensus error metric [22, 26, 31], which is computed as $\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\|\boldsymbol{w}_i^T - \bar{\boldsymbol{w}}^T\|^2$. Here, $\mathcal{B}$ is the set of benign clients, $\bar{\boldsymbol{w}}^T$ is the average of their final local models, and $\boldsymbol{w}_i^T$ is client i's final model after T training rounds in the DFL system.

Table 8: Maximum mean squared errors (Max.MSEs) of different DFL methods on synthetic dataset.

Method	No	LF	Feature	Gauss	Krum	Trim	Adapt
FedAvg	0.36	0.39	>100	>100	72.18	58.50	90.29
Krum	1.11	1.23	>100	1.19	1.18	1.18	1.19
Trim-mean	0.38	0.39	3.45	0.40	4.17	5.41	5.41
Median	0.39	0.40	2.37	0.42	1.22	3.93	3.93
FLTrust	0.41	0.46	>100	22.87	10.12	8.00	0.42
UBAR	0.40	0.40	>100	0.40	0.40	0.40	0.40
LEARN	0.42	0.42	0.42	0.64	5.36	17.78	1.60
SCCLIP	0.36	0.39	>100	0.42	5.63	5.12	4.83
BALANCE	0.36	0.36	0.36	0.36	0.36	0.36	0.36

Table 9: Consensus error of different DFL methods.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.01	0.01	>100	>100	>100	0.01	>100	>100
Krum	0.01	0.01	>100	0.01	0.01	0.01	0.01	0.01
Trim-mean	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01
Median	0.01	0.01	0.01	0.01	0.01	0.01	0.56	0.01
FLTrust	0.01	0.01	45.32	2.13	0.01	1.71	0.01	0.01
UBAR	0.01	0.01	>100	0.01	0.01	0.01	0.01	0.01
LEARN	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
SCCLIP	0.01	0.01	0.02	0.01	0.01	0.01	>100	0.01
BALANCE	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

Table 10: Results of different DFL methods, where each client aggregates its model as $\boldsymbol{w}_i^{t+1} = \mathbf{AGG}\{\boldsymbol{w}_j^{t+\frac{1}{2}}, j \in \widehat{\mathcal{N}_i}\}$, $\widehat{\mathcal{N}_i} = \mathcal{N}_i \cup \{i\}$, $\mathcal{N}_i$ is the set of neighbors of client i (not including client i itself).

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.09	0.09	0.90	0.90	0.91	0.91	0.90 / 1.00	0.90
Krum	0.18	0.18	0.13	0.18	0.22	0.18	0.18 / 0.01	0.18
Trim-mean	0.17	0.49	0.17	0.17	0.89	0.89	0.90 / 1.00	0.89
Median	0.19	0.38	0.23	0.31	0.83	0.87	0.78 / 0.01	0.88
FLTrust	0.11	0.11	0.11	0.11	0.11	0.90	0.11 / 0.90	0.12
UBAR	0.25	0.25	0.26	0.25	0.27	0.27	0.25 / 0.01	0.23
LEARN	0.10	0.15	0.11	0.10	0.35	0.55	0.10 / 0.19	0.81
SCCLIP	0.10	0.10	0.10	0.10	0.90	0.91	0.46 / 0.04	0.91
BALANCE	0.09	0.09	0.09	0.10	0.09	0.09	0.09 / 0.01	0.10

Table 11: Results of different DFL methods in Case I.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.16	0.17	0.90	0.90	0.91	0.91	0.90 / 1.00	0.90
Krum	0.23	0.24	0.90	0.26	0.26	0.28	0.26 / 0.01	0.26
Trim-mean	0.20	0.21	0.15	0.34	0.44	0.74	0.23 / 0.03	0.48
Median	0.22	0.23	0.23	0.22	0.56	0.60	0.24 / 0.02	0.33
FLTrust	0.16	0.17	0.90	0.16	0.17	0.89	0.22 / 0.16	0.91
UBAR	0.20	0.20	0.90	0.20	0.21	0.22	0.23 / 0.03	0.22
LEARN	0.16	0.21	0.25	0.17	0.30	0.64	0.16 / 0.05	0.38
SCCLIP	0.16	0.17	0.17	0.17	0.87	0.90	0.16 / 0.01	0.90
BALANCE	0.16	0.16	0.17	0.17	0.17	0.17	0.16 / 0.01	0.18

Table 12: Results of different DFL methods in Case II.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.19	0.17	0.90	0.90	0.90	0.90	0.90 / 1.00	0.90
Krum	0.21	0.22	0.90	0.27	0.25	0.26	0.26 / 0.02	0.27
Trim-mean	0.43	0.43	0.57	0.43	0.89	0.89	0.90 / 1.00	0.89
Median	0.39	0.40	0.46	0.44	0.79	0.77	0.53 / 0.05	0.77
FLTrust	0.19	0.19	0.90	0.24	0.21	0.91	0.19 / 0.86	0.90
UBAR	0.24	0.24	0.90	0.31	0.30	0.24	0.46 / 0.03	0.29
LEARN	0.32	0.35	0.32	0.32	0.75	0.89	0.33 / 0.04	0.87
SCCLIP	0.19	0.19	0.20	0.30	0.90	0.90	0.39 / 0.02	0.90
BALANCE	0.19	0.19	0.20	0.20	0.21	0.22	0.20 / 0.01	0.22

Table 13: Results of different DFL methods in Case III and Case IV.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
Case III	0.14	0.14	0.90	0.24	0.91	0.89	0.16 / 0.01	0.90
Case IV	0.17	0.19	0.90	0.09	0.75	0.90	0.23 / 0.05	0.80

Table 14: Results of different DFL methods, where clients use different initial local models.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.11	0.90	0.90	0.90	0.90	0.90 / 1.00	0.90
Krum	0.10	0.12	0.90	0.10	0.12	0.12	0.15 / 0.01	0.12
Trim-mean	0.11	0.14	0.44	0.11	0.84	0.70	0.91 / 0.01	0.59
Median	0.13	0.17	0.56	0.16	0.58	0.64	0.19 / 0.01	0.87
FLTrust	0.11	0.11	0.11	0.13	0.11	0.89	0.10 / 0.47	0.11
UBAR	0.14	0.15	0.91	0.14	0.14	0.14	0.14 / 0.01	0.16
LEARN	0.13	0.14	0.14	0.14	0.25	0.36	0.15 / 0.06	0.31
SCCLIP	0.10	0.10	0.10	0.11	0.91	0.91	0.13 / 0.01	0.91
BALANCE	0.10	0.10	0.10	0.10	0.10	0.10	0.10 / 0.01	0.11

Table 15: Results of different DFL methods on different communication graphs.

(a) Complete graph.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.10	0.90	0.90	0.91	0.91	0.90 / 1.00	0.90
Krum	0.14	0.14	0.90	0.15	0.15	0.15	0.14 / 0.01	0.15
Trim-mean	0.13	0.13	0.27	0.13	0.88	0.88	0.56 / 0.58	0.88
Median	0.13	0.16	0.46	0.22	0.89	0.88	0.20 / 0.01	0.89
FLTrust	0.11	0.11	0.11	0.13	0.11	0.91	0.10 / 0.66	0.11
UBAR	0.16	0.15	0.91	0.17	0.17	0.17	0.19 / 0.01	0.17
LEARN	0.13	0.27	0.31	0.14	0.33	0.49	0.11 / 0.02	0.44
SCCLIP	0.10	0.10	0.17	0.11	0.91	0.91	0.11 / 0.01	0.91
BALANCE	0.10	0.10	0.10	0.10	0.10	0.10	0.10 / 0.01	0.10

(b) Erdős–Rényi graph.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.10	0.90	0.91	0.90	0.87	0.90 / 1.00	0.90
Krum	0.17	0.17	0.90	0.17	0.17	0.17	0.17 / 0.01	0.90
Trim-mean	0.14	0.16	0.27	0.14	0.32	0.42	0.14 / 0.01	0.45
Median	0.17	0.22	0.43	0.24	0.30	0.65	0.17 / 0.01	0.40
FLTrust	0.10	0.10	0.11	0.11	0.09	0.90	0.10 / 0.04	0.10
UBAR	0.23	0.23	0.90	0.23	0.23	0.23	0.24 / 0.02	0.23
LEARN	0.10	0.11	0.10	0.12	0.19	0.27	0.10 / 0.01	0.12
SCCLIP	0.10	0.10	0.09	0.11	0.85	0.89	0.10 / 0.01	0.59
BALANCE	0.10	0.10	0.10	0.10	0.10	0.10	0.10 / 0.01	0.10

(c) Small-world graph.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.10	0.90	0.90	0.91	0.91	0.90 / 1.00	0.90
Krum	0.17	0.17	0.90	0.17	0.17	0.17	0.19 / 0.01	0.17
Trim-mean	0.15	0.43	0.63	0.17	0.71	0.87	0.32 / 0.01	0.80
Median	0.14	0.15	0.73	0.19	0.87	0.87	0.19 / 0.01	0.85
FLTrust	0.10	0.10	0.13	0.13	0.10	0.91	0.10 / 0.14	0.11
UBAR	0.14	0.14	0.90	0.14	0.14	0.14	0.18 / 0.01	0.14
LEARN	0.10	0.11	0.16	0.10	0.19	0.33	0.10 / 0.01	0.29
SCCLIP	0.10	0.10	0.10	0.10	0.90	0.91	0.10 / 0.01	0.49
BALANCE	0.10	0.10	0.11	0.11	0.11	0.11	0.10 / 0.01	0.11

(d) Ring graph.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.11	0.90	0.90	0.90	0.91	0.90 / 1.00	0.90
Krum	0.17	0.17	0.90	0.90	0.90	0.90	0.90 / 1.00	0.90
Trim-mean	0.19	0.21	0.90	0.90	0.90	0.90	0.90 / 1.00	0.90
Median	0.10	0.11	0.90	0.90	0.90	0.90	0.90 / 1.00	0.90
FLTrust	0.11	0.11	0.11	0.36	0.11	0.90	0.11 / 0.49	0.11
UBAR	0.13	0.13	0.90	0.13	0.13	0.13	0.27 / 0.01	0.90
LEARN	0.12	0.12	0.90	0.90	0.90	0.90	0.90 / 1.00	0.90
SCCLIP	0.11	0.11	0.13	0.11	0.83	0.90	0.11 / 0.01	0.49
BALANCE	0.10	0.10	0.11	0.10	0.10	0.10	0.11 / 0.01	0.12

Table 16: Results of different DFL methods on graphs with different FEMBs.

(a) Graph with FEMB 0.16.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.10	0.90	0.90	0.90	0.91	0.90 / 1.00	0.90
Krum	0.17	0.17	0.90	0.17	0.18	0.17	0.17 / 0.01	0.19
Trim-mean	0.13	0.13	0.13	0.13	0.31	0.39	0.13 / 0.01	0.33
Median	0.14	0.15	0.20	0.15	0.43	0.41	0.14 / 0.01	0.43
FLTrust	0.10	0.10	0.11	0.11	0.11	0.91	0.10 / 0.01	0.10
UBAR	0.17	0.17	0.90	0.18	0.18	0.18	0.17 / 0.01	0.18
LEARN	0.10	0.10	0.10	0.10	0.14	0.28	0.10 / 0.01	0.10
SCCLIP	0.10	0.10	0.10	0.11	0.81	0.91	0.10 / 0.01	0.49
BALANCE	0.10	0.10	0.10	0.10	0.10	0.10	0.10 / 0.01	0.10

(b) Graph with FEMB 0.22.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.10	0.90	0.90	0.90	0.91	0.90 / 1.00	0.90
Krum	0.17	0.17	0.90	0.17	0.17	0.17	0.17 / 0.01	0.17
Trim-mean	0.13	0.13	0.13	0.13	0.19	0.55	0.17 / 0.01	0.51
Median	0.14	0.14	0.14	0.17	0.54	0.62	0.22 / 0.01	0.59
FLTrust	0.10	0.10	0.11	0.11	0.10	0.89	0.10 / 0.04	0.10
UBAR	0.16	0.16	0.90	0.17	0.18	0.18	0.18 / 0.01	0.16
LEARN	0.10	0.13	0.12	0.10	0.12	0.43	0.10 / 0.01	0.19
SCCLIP	0.10	0.10	0.10	0.10	0.89	0.90	0.10 / 0.01	0.43
BALANCE	0.10	0.10	0.11	0.10	0.10	0.11	0.10 / 0.01	0.11

(c) Graph with FEMB 0.32.

Method	No	LF	Feature	Gauss	Krum	Trim	Backdoor	Adapt
FedAvg	0.10	0.10	0.90	0.90	0.90	0.91	0.90 / 1.00	0.90
Krum	0.17	0.17	0.90	0.17	0.19	0.17	0.17 / 0.01	0.17
Trim-mean	0.12	0.13	0.19	0.13	0.86	0.87	0.59 / 0.60	0.88
Median	0.14	0.14	0.14	0.19	0.89	0.88	0.86 / 0.01	0.89
FLTrust	0.10	0.10	0.12	0.12	0.10	0.91	0.10 / 0.01	0.11
UBAR	0.16	0.17	0.91	0.17	0.19	0.19	0.21 / 0.01	0.28
LEARN	0.13	0.13	0.25	0.13	0.19	0.43	0.13 / 0.01	0.39
SCCLIP	0.10	0.10	0.10	0.10	0.90	0.91	0.10 / 0.01	0.59
BALANCE	0.10	0.10	0.10	0.10	0.10	0.10	0.10 / 0.01	0.10