LiD-FL: Towards List-Decodable Federated Learning

Hong Liu[1], Liren Shan[2], Han Bao[1], Ronghui You[3*], Yuhao Yi[1,4*], Jiancheng Lv[1]

[1]College of Computer Science, Sichuan University
[2]Toyota Technological Institute at Chicago
[3]School of Statistics and Data Science, Nankai University
[4]Institute of Clinical Pathology, West China Hospital, Sichuan University
hong_liu@stu.scu.edu.cn, lirenshan@ttic.edu, baohan1@stu.scu.edu.cn, yourh@nankai.edu.cn, yuhaoyi@scu.edu.cn,
lvjiancheng@scu.edu.cn

Abstract

Federated learning is often used in environments with many unverified participants. Therefore, federated learning under adversarial attacks receives significant attention. This paper proposes an algorithmic framework for list-decodable federated learning, where a central server maintains a list of models, with at least one guaranteed to perform well. The framework has no strict restriction on the fraction of honest clients, extending the applicability of Byzantine federated learning to the scenario with more than half adversaries. Assuming the variance of gradient noise in stochastic gradient descent is bounded, we prove a convergence theorem of our method for strongly convex and smooth losses. Experimental results, including image classification tasks with both convex and non-convex losses, demonstrate that the proposed algorithm can withstand the malicious majority under various attacks.

1   Introduction

Federated learning (FL) has gained popularity in distributed learning due to its nature of keeping the data decentralized and private (Kairouz et al. 2021). With significant scalability, FL is applied to a number of large-scale distributed systems, such as finance and healthcare software (Yang et al. 2019; Nguyen et al. 2023). In a typical federated learning setting, multiple devices (called *clients*) collaboratively train a global model, each of which preserves a chunk of the training data. FL employs a central server (also called *parameter server*), which holds no data, to optimize the global model iteratively by aggregating local model updates from clients (McMahan et al. 2017; Bonawitz et al. 2019).

Although the federated learning paradigm improves privacy and the overall training efficiency, its training procedure is prone to malicious attacks (Karimireddy, He, and Jaggi 2022; Farhadkhani et al. 2024). Byzantine clients refers to arbitrarily behaving faulty clients except for changing the identities and behaviors of other clients. A variety of instantiations of the Byzantine attack model are designed, with the goal of preventing the model from converging or creating a backdoor to the system (Bagdasaryan et al. 2020; Shejwalkar et al. 2022). Previous study has shown that well-designed Byzantine attacks could degrade the global model

or lead it to collapse (Blanchard et al. 2017; Shejwalkar and Houmansadr 2021; Baruch, Baruch, and Goldberg 2019; Xie, Koyejo, and Gupta 2020; Fang et al. 2020).

To address Byzantine attacks, a number of Byzantine-robust FL algorithms have been proposed. These algorithms usually leverage robust aggregation rules to filter out abnormal clients or control the influence of Byzantine clients to acceptable level (Blanchard et al. 2017; Pillutla, Kakade, and Harchaoui 2022a; Karimireddy, He, and Jaggi 2021; El Mhamdi, Guerraoui, and Rouault 2018; Allouah et al. 2023; Yi et al. 2024; Luan et al. 2024). The proposed Byzantine-robust FL algorithms are investigated from various perspectives, advancing both theoretical foundations and practical frontiers of this theme. It has been shown that previous solutions which use robust aggregators to train a global model have a breaking point of $1/2$ (Gupta and Vaidya 2020; Yin et al. 2018; Allouah et al. 2023; Farhadkhani et al. 2022; Allouah et al. 2024b). When the honest clients are less than one half, the robust aggregation rules lose efficacy.

However, the assumption of an honest majority does not necessarily hold under Sybil attacks, in which the attacker creates massive pseudonymous identities. Therefore it is important to consider adversarial scenarios where over half of the clients are malicious. Prior work considering more than half malicious clients includes the FLTrust algorithm (Cao et al. 2021) and the RDGD algorithm (Zhao and Wan 2024), which resort to a small verified dataset on the server to push the breaking point over $1/2$. However, such assumption is unavailable for privacy sensitive federated learning scenarios. Additionally, the performance of the global model significantly relies on the availability of an adequately representative validation set on the server, which can be challenging to gather.

In this paper, we propose a list-decodable federated learning framework, LiD-FL. Conventional robust federated learning algorithms often rely on resilient aggregation rules to defend the learning process and optimize a single global model iteratively. Unlike previous Byzantine-robust FL schemes, LiD-FL preserves a list of global models and optimizes them concurrently. In each global round, LiD-FL randomly samples a global model from the list, which promises a constant probability of obtaining the valid model in the list. Further, the server attains a model update from the

---

clients using a randomized protocol, with a positive probability of obtaining non-Byzantine updates. Then we employ a voting procedure to update the model list, ensuring that at least one of the elected models is not poisoned by Byzantine clients. Therefore, the algorithm guarantees that there exits one valid model in the list, which is optimized at a proper speed with a constant probability. In the case where the majority is honest our framework degenerates into a Byzantine-robust FL solution with a list size of one.

Main contributions. The main contributions of this paper are as follows.

1) We propose a Byzantine-robust FL framework, LiD-FL, designed to overcome the limitations of existing robust FL methods, which break down when honest clients constitute less than half of the total. LiD-FL imposes no restriction on the ratio of honest clients, as long as a lower bound of the ratio is known to the server. By providing a list of models wherein at least one is guaranteed to be acceptable, LiD-FL expands the applicability of Byzantine federated learning to environments with an untrusted majority.

2) We show a convergence theorem for the proposed algorithm when a strongly convex and smooth loss function is applied and the local data of clients are independently and identically distributed (i.i.d.).

3) We evaluate the performance of the proposed algorithm by running experiments for image classification tasks using a variety of models with both convex and non-convex loss functions, including logistic regression (LR) and convolutional neural networks (CNN). The results demonstrate the effectiveness, robustness and stability of LiD-FL.

Related work. Although the emergence of FL has alleviated the data fragmentation issue to a certain degree (Konečný et al. 2016; McMahan et al. 2017), it is known to be vulnerable to device failure, communication breakdown and malicious attacks (El Mhamdi, Guerraoui, and Rouault 2018). In recent years, algorithmic frameworks are proposed to enhance the robustness of FL towards unreliable clients (Yin et al. 2018; Liu, Gupta, and Vaidya 2021; Allen-Zhu et al. 2021). Farhadkhani et al. propose a unified framework for Byzantine-robust distributed learning, RESAM, which utilizes resilient aggregation and distributed momentum to enhance the robustness of learning process.

One of the key ingredients of prior robust FL algorithms is the Byzantine resilient aggregation rules, also called robust aggregators. The federated averaging algorithm (McMahan et al. 2017) uses naive average of local models to update the global model, which is effective in non-adversarial scenarios but could be easily destroyed by a single malicious client (Blanchard et al. 2017). Yin et al. propose two robust aggregations rules based on coordinate median and trimmed mean (TM), respectively. The theoretical analysis proves their statistical error rates for both convex and non-convex optimization objective, while their statistical rates for strongly convex losses are order-optimal (Yin et al. 2018). Geometric median (GM) has been proved to be an appropriate approximation of average as well as robust to adversarial clients (El-Mhamdi et al. 2023). The RFA algorithm leverages an approximate algorithm of GM to defend the federated learning against Byzantine attacks, which is computationally efficient and numerically stable (Pillutla, Kakade, and Harchaoui 2022a).

Beside the statistical methods, many robust aggregators based on certain criteria of local updates are developed. The ARC (Allouah et al. 2024a), CClip (Karimireddy, He, and Jaggi 2021) and Norm (Sun et al. 2019) methods conduct norm prune on local model updates to resist Byzantine attacks, with a pre-computed of adaptive norm bound.

The parameter server is assumed to hold no data in most of previous work about Byzantine-robust FL. The above algorithms, except for TM, do not demand prior knowledge about the fraction of Byzantine clients as input. However all of them require a Byzantine fraction of less than $1/2$ to ensure robustness (Gupta and Vaidya 2020; Yin et al. 2018). Limited work has been done to address the 51% attack, where the Byzantine clients outnumber the honest clients. The FLTrust algorithm (Cao et al. 2021) assumes that the server is accessible to a tiny, clean dataset, which could be used to validate the local updates. Specifically, FLTrust weights local updates based on their cosine similarity to the model update on server dataset. Although FLTrust could tolerate more than half Byzantine clients, it is inapplicable when public dataset is unavailable.

The list-decoding technique was first introduced to the construction of error correcting codes and has then found applications in learning. A List-decoding model was introduced in (Balcan, Blum, and Vempala 2008) to solve the problem of finding proper properties of similarity functions for high-quality clustering. List-decodable learning is later proven to be an effective paradigm for the estimation, learning, and optimization tasks based on untrusted data (Charikar, Steinhardt, and Valiant 2017; Diakonikolas, Kane, and Stewart 2018). Karmalkar, Klivans, and Kothari propose a list-decodable learning algorithm for outlier-robust linear regression, with an arbitrary fraction of hostile outliers.

These studies show the usefulness of list-decodable learning for handling high-dimensional data and enhancing robust learning, even with a significant corruption ratio. The key idea of list-decodable learning is to generate a list of $\text{poly}(\frac{1}{\gamma})$ possible answers including at least one correct answer, where γ is the proportion of valid data. In this paper, we adapt the framework of list-decodable learning to Byzantine-robust federated learning.

Outline. The remainder of the paper is organized as follows: In Section 2 we introduce the robust federated learning scheme and basic notations. In Section 3 we state the proposed LiD-FL algorithm in detail. In Section 4, we provide theoretical analyses on LiD-FL and prove its convergence for strongly convex and smooth optimization objective. In Section 5 we show the empirical results, followed by the section for conclusion and future work.

2 Preliminaries

In this section, we first provide the conventional federated learning framework and then describe a Byzantine-robust federated learning setting.

2.1 Problem Setup

We consider a server-client federated learning system with one central server and m clients $C = \{c_1, \ldots, c_m\}$. The server has no data and each client holds a block of data. The set of all training data is denoted by $Z = \{(x_i, y_i)\}_{i=1}^n$, where (x_i, y_i) represents an input-output pair. The indices of training data for each client $c_j, j \in [m]$ is $D_j \subseteq [n]$. We consider the homogeneous setting in which the data possessed by every client are independently and identically sampled from the same distribution $\mathcal{D}$.

For each input-output pair (x_i, y_i), let $f_i : \mathbb{R}^d \times Z \to \mathbb{R}$ be its loss function. Given a model with parameter $w \in \mathbb{R}^d$, the loss of this model at (x_i, y_i) is $f_i(w)$. We assume $f_i(w)$ is almost everywhere differentiable with respect to the parameter. Following the definition in (Allouah et al. 2024a), we define the local loss function for client c_j as $f^{(j)}(w) = \frac{1}{|D_j|} \sum_{i \in D_j} f_i(w)$. The global loss function is the average of local loss functions $f(w) = \frac{1}{m} \sum_{j \in [m]} f^{(j)}(w)$. Then, our goal is to find the model parameter $w^* \overset{\text{def}}{=} \arg\min_{w \in \mathbb{R}^d} f(w)$ to minimize the global loss.

2.2 Byzantine-Robust Federated Learning

Federated learning employs distributed stochastic gradient descent method to solve the above optimization problem. Specifically, each client holds a local model on its private dataset, and the server maintains a global model via aggregating the local models. The global model is optimized through stochastic gradient descent (SGD) iteratively. For each global iteration $t \in \{0, \ldots, T-1\}$, the workflow of federated learning includes the following three steps:

(1) **Server Broadcast**: the server samples a subset of clients $S_t \subseteq C$ and sends the current global model $\tilde{w}_t$ to them.

(2) **Local Training**: Each sampled client $c_j \in S_t$ initially sets the local model $w_0^{(j)}$ as $\tilde{w}_t$, and optimizes it via SGD for τ local batches. Then c_j sends the local model update $u_t^{(j)} = w_\tau^{(j)} - w_0^{(j)}$ to the server.

(3) **Global Update**: the server aggregates received local model updates $U_t = \{u_t^{(j)}\}_{j \in S_t}$ via an aggregation rule $h : \mathbb{R}^{d \times |S_t|} \to \mathbb{R}^d$, e.g., taking the average, then updates the global model with aggregated update $w_{t+1} = w_t + h(U_t)$.

Now we look at a typical adversarial scenario where the server is reliable and k out of m clients are honest (or non-Byzantine) while the remainder are Byzantine (Lamport, Shostak, and Pease 2019). We use $\mathcal{H}, \mathcal{B} \subseteq [m]$ to depict the indices of honest clients and Byzantine clients, respectively. Thus the fraction of honest clients is $\gamma = k/m$. The Byzantine clients are free to disregard the prescribed learning protocol and behave arbitrarily. In particular, they send malicious local model updates to the server in the training process, trying to destroy the global model. Nonetheless, they are prohibited from corrupting other clients, altering the message of any other nodes, or hindering the transmission of messages between the server and any honest clients.

Therefore, in the Byzantine-robust federated learning, we define the global loss function as the average of local loss functions over all honest clients $f(w) = \frac{1}{|\mathcal{H}|} \sum_{j \in \mathcal{H}} f^{(j)}(w)$. Then, we aim to find the model parameter $w^* = \arg\min_w f(w)$ to minimize this global loss, with only a γ fraction of reliable clients in the system (Karimireddy, He, and Jaggi 2022).

3 List-Decodable Federated Learning

In this section, we propose a list-decodable federated learning framework named LiD-FL. It is a two-phase framework. The first phase is similar to the typical workflow of federated learning except that the server holds a list of global models. And the second phase conducts the list updating through a voting procedure. Specifically, LiD-FL differs from the conventional robust FL in Section 2 from three aspects: (i) LiD-FL maintains a list of global models L instead of a single global model. In particular, LiD-FL preserves q global models on the server and optimizes them concurrently. (ii) LiD-FL replaces the aggregator with a random sampler to update the global model. Since existing aggregators can not promise attaining honest updates with more than $1/2$ fraction of Byzantine clients, we randomly sample one client for local training and global update. (iii) LiD-FL introduces a voting procedure to update the global model list, which ensures that at least one model in the list is valid.

The whole process of LiD-FL is described as follows. We omit the procedures the same as that in Section 2 for simplicity. For each training round t:

Step 1. Server Broadcast: the server randomly chooses a global model $\tilde{w}_t$ from the current list L_t and sends it to a sampled client. We note that only one client $c_j \in C$ is sampled *uniformly at random* in each round.

Step 2. Local Training: the sampled client c_j optimizes the global model using its local dataset, then sends the model update $u_t^{(j)}$ to the server. If c_j is a Byzantine client, it sends arbitrary information to the server[1].

Step 3. Global Update: the server calculates a new global model using the received update $w_t' = w_t + u_t^{(j)}$, and adds the new model into the global model list.

Step 4. Local Voting: The server broadcasts the list L_t to all clients. Every client votes for one model in L_t. Each honest client $c_j, j \in \mathcal{H}$ evaluates the loss of all global models via a validation oracle. It then votes for the model with the minimum loss value. Byzantine clients vote arbitrarily.

Step 5. List Update: The server discards the model which receives the least number of votes to obtain a new list L_{t+1}.

Steps 1-5 are conducted repeatedly for T global rounds, then LiD-FL outputs the global model list L_T. Although not all models in the final list are valid, the clients could deploy the model they consider to be acceptable by evaluating models in the list using their local dataset. Algorithm 1 shows the pseudocode of LiD-FL.

Remark I. Guarantee of the voting procedure. In the voting procedure, each client casts one vote among the

---

[1]If a Byzantine client does not respond, the server can simply ignore it or fake an arbitrary update for it. The same strategy can be applied to the voting step.

Algorithm 1: LiD-FL

Initialization: For the server, the global model list L_0, the number of global iterations T; for each client c_j: the number of local training rounds τ, batch size b, learning rate ℓ.

1: **for** $t \in \{0, \dots, T-1\}$ **do**
2:    **Server** randomly samples one client $c_j \in C$ and one global model $\tilde{w}_t \in L_t$, then sends $\tilde{w}_t$ to c_j.
3:    **if** $j \in \mathcal{H}$ **then**
4:       $w_0^{(j)} \leftarrow \tilde{w}_t$.
5:       **for** $r \in \{0, \dots, \tau-1\}$ **do**
6:          **Client** c_j computes a stochastic gradient $g_r^{(j)}$ on a batch of local data.
7:          Update the local model:

$$w_{r+1}^{(j)} \leftarrow w_r^{(j)} - \ell \cdot g_r^{(j)}.$$

8:       **end for**
9:       $u_t^{(j)} \leftarrow w_\tau^{(j)} - w_0^{(j)}$.
10:      c_j sends $u_t^{(j)}$ to the server.
11:    **end if**
12:    *(A Byzantine client c_j sends arbitrary update $\tilde{u}_t^{(j)}$ to the server.)*
13:    **Server** extends the global model list:

$$\tilde{w}_t' \leftarrow \tilde{w}_t + u_t^{(j)},$$
$$L_t \leftarrow L_t \cup \{\tilde{w}_t'\}.$$

14:    **Server** broadcasts L_t to all clients.
15:    **for** each honest client c_j, $j \in \mathcal{H}$, in parallel **do**
16:       Evaluate the lo ss of all models using its validation set, and send the index of the model with lowest loss to the server.
17:    **end for**
18:    *(A Byzantine client sends an arbitrary index to the server.)*
19:    Server removes the model $\hat{w}_t$ which receives the least number of votes from the list:

$$L_{t+1} \leftarrow L_t \backslash \{\hat{w}_t\}.$$

20: **end for**
21: **return** L_T.

$(q+1)$ candidate models. Then the top q candidates with the highest number of votes are preserved by the server. To ensure that at least one model voted by honest clients is maintained, the size of the global model list q must satisfy $q \geq \lfloor 1/\gamma \rfloor$. We compare the efficacy of LiD-FL with different values of q in the appendix, and the results show that when the bound is satisfied, a larger list size attains better stability while maintaining comparable accuracy.

Remark II. Applying aggregation rules in LiD-FL. We use a simple random sampler to obtain local updates, which is valid and computationally efficient. However, the update of only one client is collected in each global round, which does not utilize the parallel computation power of clients. A more efficient way is to use robust aggregation rules to

generate a list of aggregated updates. A preliminary experimental result is given in the appendix, which shows that by leveraging aggregation rules, the algorithm performs better.

4 Analysis

In this section, we show that under proper assumptions, our algorithm converges to the optimal parameters with a large success probability. We use $Z' = \bigcup_{j \in \mathcal{H}} \{(x_i, y_i), i \in D_j\}$ to denote the set of all training data possessed by honest clients. We assume that the loss function f_i for each input-output pair (x_i, y_i) is α-strongly convex and β-smooth. The variance of the gradient noise in SGD is assumed to be bounded by σ^2. All assumptions are declared formally in the appendix. Let w^* be the optimal parameter for the model that minimizes the global loss function f. To simplify the analysis, we let $\tau = 1$, $\ell = 1/\beta$ and $|D_j| = n/m$ for all $j \in [m]$. With this balanced dataset assumption, the global loss is also the empirical loss on Z', i.e., $f(w) = \frac{1}{|Z'|} \sum_{(x_i, y_i) \in Z'} f_i(w)$. This balanced dataset assumption can be removed by normalizing the gradient update in the algorithm.

We assume that each client has access to a validation oracle. This oracle takes the parameter w of a model as input and then outputs the estimation of total loss given by this model w.

Definition 4.1. *A validation oracle is (η, p)-accurate if for any parameter w, with probability at least p, its estimation $\hat{f}(w)$ of total loss at w satisfies*

$$|\hat{f}(w) - f(w)| \leq \eta.$$

We assume that each client independently gets an accurate estimation through such an oracle. Our analysis can be easily extended to the case where the validation processes between clients are dependent. We give an example of a validation oracle in the appendix.

Theorem 4.2. *Suppose γ fraction of all m clients are uncorrupted. Assume each loss function f_i is α-stronly convex and β-smooth, and the variance of the gradient noise on each client is bounded by σ^2. With a list size of $q \geq \lfloor 1/\gamma \rfloor$, Algorithm 1 guarantees that after T rounds, for all $m \geq \ln(1/\delta)/(2(\gamma p^{q+1} - 1/((q+1))))$, there is one model with parameter w_T in the output such that*

$$\mathbf{E}[f(w_T) - f(w^*)] \leq e^{-\frac{\alpha \gamma (1-\delta)}{\beta q} \cdot T} \mathbf{E}[f(w_0) - f(w^*)] + \varepsilon$$

holds with probability at least $1 - \delta T$, where $\varepsilon = q(1 - \delta)(4\eta\beta + \sigma^2)/(2\alpha\gamma)$.

We first define some useful notations. For each round t, we use w_t to denote the parameter of the model that achieves the smallest total loss among the model list maintained by Algorithm 1 after this round. Let w_0 be the initial model parameter. We divide each round into two cases based on random decisions made by Algorithm 1 at round t: (1) good round; (2) bad round. We call this round t a *good* round if Algorithm 1 chooses the best model with parameter w_{t-1} maintained from the last round and samples the update provided by an honest client. Otherwise, we call this round a

bad round. Let G_t, B_t be the event that round t is good, and the event that round t is bad, respectively. Consider the voting process at round t. Let w', w'_1, w'_2, ..., w'_q be all candidate models in the voting process, where w' is the best model among all candidates. We say the voting process has a *failure* if the following two conditions are satisfied: (1) the loss of w' is at least 2η separated from the loss of other candidate models, $f(w'_i) \geq f(w') + 2\eta$ for all $i = 1, 2, \ldots, q$; and (2) the model w' is not chosen by the voting process. Let F_t be the event that a failure happens at round t and F_t^c be its complement.

Lemma 4.3. *For each round t, the failure happens with probability at most δ.*

Proof. Let all candidate models in the voting process are w', w'_1, w'_2, ..., w'_q. Suppose for all $i = 1, 2, \ldots, q$, the loss of model w'_i is at least $f(w') + 2\eta$, where η is the accuracy of the validation oracle used in Algorithm 1. If the round t does not satisfy these conditions, then by definition, the failure will not happen.

Now, we bound the probability that the model w' is not chosen by the voting process, which implies a failure happens. Each honest client will query a (η, p)-accurate validation oracle to estimate the loss of all $q + 1$ candidate parameters. Note that the suboptimal models $w'_1, w'_2, \ldots, w'_q$ have a total loss at least $f(w') + 2\eta$, which is 2η separated from the loss of the best model w' among candidates. Thus, with probability at least p^{q+1}, the loss estimation of w' is smaller than the loss estimations of all other candidates given by the validation oracle. Then, with probability at least p^{q+1}, an honest client will vote for the model w'. The total number of clients is m and the number of honest clients is $k = \gamma m$. If more than $m/(q + 1)$ honest clients vote for parameter w', then the parameter w' will be chosen by the voting process. Let Y_i be the event that the honest client i votes for w'. By Hoeffding's inequality, the probability that the parameter w' is not chosen is at most

$$\Pr\left\{ \sum_{i=1}^{k} Y_i \leq \frac{m}{q+1} \right\}$$
$$\leq \exp\left(-\frac{2(p^{q+1}k - m/(q+1))^2}{k} \right)$$
$$\leq e^{-2(p^{q+1} - 1/((q+1)\gamma))k} \leq \delta,$$

where the last inequality is due to $p^{q+1} > 1/((q + 1)\gamma)$ and $k \geq \ln(1/\delta)/(2(p^{q+1} - 1/((q + 1)\gamma)))$. $\square$

In the following analysis, we assume that the failure does not happen at each round. Let G'_t be the event that a good round without a failure and B'_t be the event that a bad round without a failure, respectively. We show the following recursive relation.

Lemma 4.4. *Let $\check{\eta} = 2\eta + \sigma^2/(2\beta)$, for each round $t \geq 1$, we have*

$$\mathbf{E}[(f(w_t) - f(w^*)) \mid F_t^c] \leq$$
$$\leq \left(1 - \frac{\alpha\gamma}{\beta q}(1 - \delta) \right) \mathbf{E}[f(w_{t-1}) - f(w^*)] + \check{\eta},$$

Proof. We analyze the good round and bad round separately.

Good Round: We first consider a good round. Let w' be the updated parameter sampled by the algorithm. Conditioned on the good round, the updated parameter follows a stochastic gradient descent on all data possessed by honest clients Z'. Since f is the sum of α-convex and β-smooth functions, we have

$$\mathbf{E}_{w'}[f(w') - f(w^*) \mid G_t] \leq$$
$$\leq \left(1 - \frac{\alpha}{\beta} \right) \cdot \mathbf{E}_{w'}[(f(w_{t-1}) - f(w^*))] + \sigma^2/(2\beta).$$

Now, we consider the voting process at a good round t. The updated model w' is among the $q + 1$ candidate models in the voting process. Note that w_t is the best model among all models maintained after round t. Suppose the voting process has no failure. Then we have either the model w' is chosen by the voting process or another candidate model w'' among $w'_1, w'_2, \ldots, w'_q$ has a loss at most $f(w') + 2\eta$. In the latter case, at least one of w' and w'' is chosen by the voting process. Thus, we have $f(w_t) \leq f(w') + 2\eta$. For a good round t without a failure, we have

$$\mathbf{E}_{w_t}[f(w_t) - f(w^*) \mid G'_t] \leq$$
$$\leq \left(1 - \frac{\alpha}{\beta} \right) \cdot \mathbf{E}_{w_t}[(f(w_{t-1}) - f(w^*))] + \check{\eta}.$$

Bad Round: We now consider a bad round t. Let w_{t-1} and $w'_1, w'_2, \ldots, w'_q$ be all candidate models in this case. Since the failure does not happen, we have either w_{t-1} is chosen by the voting process, or one model among $w'_1, w'_2, \ldots, w'_q$ has a loss at most $f(w_{t-1}) + 2\eta$. Thus, we have

$$\mathbf{E}_{w_t}[f(w_t) - f(w^*) \mid B'_t] \leq \mathbf{E}_{w_t}[f(w_{t-1}) - f(w^*)] + 2\eta.$$

Now, we combine two cases. Note that

$$\mathbf{E}_{w_t}[(f(w_t) - f(w^*)) \mid F_t^c\}] \Pr\{F_t^c\} \leq$$
$$\leq \mathbf{E}_{w_t}[f(w_t) - f(w^*) \mid G'_t] \Pr\{G'_t\}$$
$$+ \mathbf{E}_{w_t}[f(w_t) - f(w^*) \mid B'_t] \Pr\{B'_t\}.$$

The good round G_t happens with probability at least γ/q since the algorithm samples the best model with probability at least $1/q$ and an update from an honest client with probability γ. Similar to the analysis in Lemma 4.3, conditioned on a good round G_t, the failure still happens with probability at most δ. Thus, we have

$$\Pr\{G'_t \mid F_t^c\} = \frac{\Pr\{G_t, F_t^c\}}{\Pr\{F_t^c\}} \geq \Pr\{G_t\} \Pr\{G_t \mid F_t^c\}$$
$$\geq \frac{\gamma}{q}(1 - \delta).$$

Thus, we have

$$\mathbf{E}_{w_t}[(f(w_t) - f(w^*)) \mid F_t^c\}] \leq$$
$$\leq \left(1 - \Pr\{G'_t \mid F_t^c\} \cdot \frac{\alpha}{\beta} \right) \mathbf{E}_{w_t}[(f(w_{t-1}) - f(w^*))] + \check{\eta}$$
$$\leq \left(1 - \frac{\alpha\gamma}{\beta q}(1 - \delta) \right) \mathbf{E}_{w_t}[(f(w_{t-1}) - f(w^*))] + \check{\eta}.$$

$\square$

Now, we prove the main theorem.

Proof of Theorem 4.2. We consider the loss of the final model w_T when there is no failure among all T rounds. Let F be the event that there is a failure happens among T rounds, and F^c be its complement. By Lemma 4.3 and the union bound, the event F^c happens with probability at least $1 - \delta T$.

Now, we analyze the expected loss of the output model conditioned on no failure happening in all T rounds. By Lemma 4.4, we have for any round $t \leq T$,

$$\mathbf{E}[(f(w_t) - f(w^*)) \mid F_t^c] \leq$$
$$\leq \left(1 - \frac{\alpha\gamma}{\beta q}(1 - \delta)\right) \mathbf{E}[f(w_{t-1}) - f(w^*)] + \check{\eta}.$$

Thus, we have

$$\mathbf{E}[(f(w_T) - f(w^*)) \mid F^c] \leq$$
$$\leq \left(1 - \frac{\alpha\gamma}{\beta q}(1 - \delta)\right)^T \mathbf{E}[f(w_0) - f(w^*)]$$
$$+ \check{\eta} \cdot (1 - \delta)\frac{\beta q}{\alpha\gamma}.$$

By taking $\check{\eta} = \frac{\varepsilon\alpha\gamma}{\beta q(1-\delta)}$, we get the conclusion. $\square$

5 Experiments

In this section, we evaluate our method on image classification tasks with both convex and non-convex optimization objectives. Limited by space, we leave additional implementation details and full results to the appendix.

5.1 Experimental Setup

Dataset. We conduct experiments on two datasets, FEMNIST (Caldas et al. 2019) and CIFAR-10 (Krizhevsky 2009). The dataset introduction and data distribution are as follows:

1. **FEMNIST** Federated Extended MNIST (FEMNIST) is a standard benchmark for federated learning, which is built by partitioning data in the EMNIST (Cohen et al. 2017) dataset. The FEMNIST dataset includes 805,263 images across 62 classes. We sample 5% of the images from the original dataset and distribute them to clients in an i.i.d manner. Note that we simulate an imbalanced data distribution, while the number of training samples differs among clients. The implementation is based on LEAF [2]. Furthermore, we split the local data on each client into a training set, a validation set and a test set with a ratio of $36 : 9 : 5$.

2. **CIFAR-10** The CIFAR-10 dataset consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. We sample $5/6$ of the samples from original dataset *uniformly at random* to construct our dataset. The sampled images are evenly distributed to clients. For each client, we split the local data into a training set, a validation set and a test set with a ratio of $4 : 1 : 1$.

Byzantine attacks. Byzantine clients send elaborate malicious updates to the server. In LiD-FL, the server samples only one client per iteration to perform local training and obtain a model update from that client. When a Byzantine client is chosen, the honest clients do not launch local training. However, some attacks fabricate malicious updates based on the mean and variance of honest updates. In this scenario, we model an omniscient attacker (Baruch, Baruch, and Goldberg 2019), who knows the data of all clients. Thus it can mimic the statistical properties of honest updates and then craft a malicious update.

We simulate one data poison attack, the data label flipping attack (**LF**) and 5 model poison attacks: the inner product manipulation attack (**EPR**) (Xie, Koyejo, and Gupta 2020), the random Gauss vector attack (**Gauss**), the little perturbations attack (**LIE**) (Baruch, Baruch, and Goldberg 2019), the Omniscient attack (**OMN**) (Blanchard et al. 2017), and the sign flipping attack (**SF**). Refer to the appendix for detailed attack configurations.

As we introduce a voting procedure into the training process, we design two types of voting attack for Byzantine clients: (1) **Worst**: voting for the model with the highest validation loss; (2) **Random**: casting a vote randomly among the q models with highest validation losses, i.e., excluding the model with the lowest validation loss.

Baselines. We compare the proposed LiD-FL algorithm with 4 prior robust FL algorithms: naive average (**FedAvg**) (McMahan et al. 2017), geometric median approximated by the Weiszfeld's algorithm (**GM**) (Pillutla, Kakade, and Harchaoui 2022b) with the 1 iteration, norm bounded mean (**Norm**) (Sun et al. 2019) with hyperparameter $\tau = 0.215771$, and coordinate-wise median (**CWM**) (Yin et al. 2018). In experiments we incorporate the notion of momentum in local updates. We denote the initial momentum and momentum coefficient of client c_j by $v_0^{(j)}$ and μ, respectively. The local training step with momentum is given by $v_{r+1}^{(j)} \leftarrow \mu \cdot v_r^{(j)} + g_r^{(j)}$, followed by $w_{r+1}^{(j)} \leftarrow w_r^{(j)} - \ell \cdot v_{r+1}^{(j)}$.

Models and parameters. For both datasets, we simulate a distributed system with 1 server and 35 clients. We perform experiments for 3 levels of Byzantine fraction: 0.4, 0.6, and 0.8. For the FEMNIST dataset, we train a logistic regression (LR) classifier for convex optimization and a convolutional neural network (CNN) for non-convex optimization, respectively. For the CIFAR-10 dataset, we train a CNN model. Detailed model architecture and hyperparameter settings are deferred to the appendix.

Once the training process is complete, each client tests all models in the global list using its local test dataset. We then calculate the average test accuracy across all clients for each model and present the highest accuracy as the test result. We repeat each experiment for 5 times, each with a different random seed, and report the average test result across the 5 runs [3]. The implementation is based on PRIOR[4].

5.2 Analysis of Results

Method	EPR	Gauss	LF	LIE	OMN	SF	Wst
CWM	0.02	0.58	0.04	0.04	0.02	0.03	0.02
FedAvg	0.02	**0.62**	0.09	0.06	0.02	0.02	0.02
GM	0.02	0.61	0.05	0.05	0.02	0.02	0.02
Norm	0.01	0.61	0.02	0.44	0.01	0.01	0.01
LiD-FL	**0.57**	0.55	**0.56**	**0.57**	**0.57**	**0.56**	**0.55**

Table 1: Performance comparison on FEMNIST at a Byzantine fraction of 0.6, with the LR global model.

Method	EPR	Gauss	LF	LIE	OMN	SF	Wst
CWM	0.05	0.34	0.14	0.01	0.05	0.05	0.01
FedAvg	0.05	0.33	0.24	0.01	0.05	0.05	0.01
GM	0.05	0.19	0.33	0.01	0.05	0.05	0.01
Norm	0.00	**0.78**	0.06	0.05	0.00	0.05	0.00
LiD-FL	**0.72**	0.71	**0.72**	**0.72**	**0.73**	**0.73**	**0.71**

Table 2: Performance comparison on FEMNIST at a Byzantine fraction of 0.6, with the CNN global model.

Table 1 and Table 2 show the test accuracy of various algorithms on FEMNIST at a Byzantine fraction of 0.6 with the LR model and the CNN model, respectively. The last column displays the worst-case among the 6 attacks. We use the Worst voting attack unless otherwise noted.

The results demonstrate the remarkable effectiveness of LiD-FL, as its worst test accuracy across different attacks is the highest on both LR model and CNN model while all the baseline methods collapsed. Specifically, for the LR model, the worst test accuracy of LiD-FL is 0.56, and the other methods fail to produce a valid model. For the CNN model, none of the baselines successfully defend the learning process against the EPR, LF, LIE, OMN and SF attacks, while LiD-FL achieves a test accuracy over 72% against these 5 attacks. For the Gauss attack, the Norm shows best performance and LiD-FL achieves the second highest test accuracy. Additionally, the performance of LiD-FL varies slightly across diverse attacks on both models, indicating the excellent robustness of LiD-FL.

Figure 1 illustrates the experimental results on CIFAR-10 at a Byzantine fraction of 0.6. The results emphasize that when exposed to EPR, LF, LIE, OMN and SF attacks, LiD-FL much outperforms other methods. The test accuracy of LiD-FL is more than 0.50 for these attacks, while the test accuracy of other algorithms all remain below 0.3. Note that under the Gauss attack, the LiD-FL shows a lower convergence speed than baselines.

Voting attacks. Table 3 presents the test accuracy of LiD-FL under two voting attacks, where "W" and "R" represent the Worst voting attack and Random voting attack, respectively. LiD-FL achieves consistently high performance
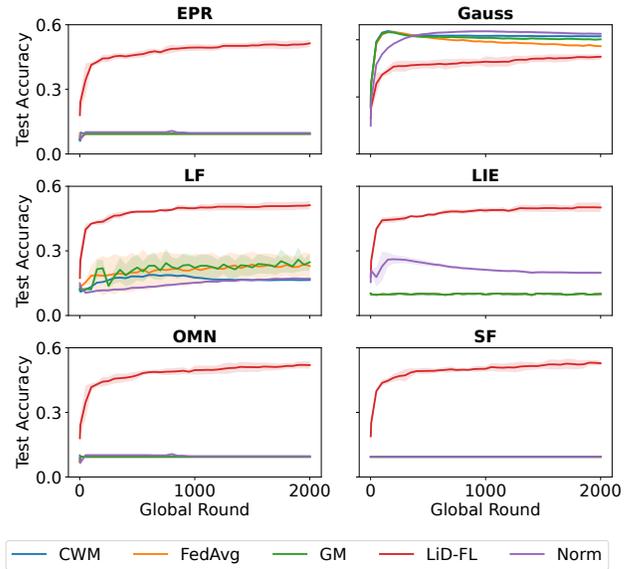

Figure 1: Test accuracy of different methods on CIFAR-10 at a Byzantine fraction of 0.6, with the CNN global model.

against both voting attacks, demonstrating remarkable robustness. In most cases, the Random voting attack is less effective than the Worst voting attack. This is because when the new model is suboptimal but not poisoned, the Random voting scheme increases its chance of being optimized.

Model	ATK	EPR	Gauss	LF	LIE	OMN	SF	Wst
LR	W	0.57	0.55	0.56	0.57	0.57	0.56	0.55
	R	0.56	0.56	0.56	0.58	0.57	0.57	0.56
CNN	W	0.72	0.71	0.72	0.72	0.73	0.73	0.71
	R	0.73	0.69	0.73	0.60	0.73	0.75	0.60

Table 3: Test accuracy of LiD-FL under different voting attacks on FEMNIST at a Byzantine fraction of 0.6.

6 Conclusion

In this paper, we propose a list decodable federated learning framework called LiD-FL. Unlike traditional robust FL algorithms, LiD-FL maintains a list of global models on the server, ensuring that at least one of them is reliable in adversarial environments. One significant advantage of LiD-FL is that it has no strict restriction on the fraction of malicious clients. Our theoretical analysis proves a convergence theorem for strongly convex and smooth optimization objectives. Numerical simulations on two datasets demonstrate the effectiveness, robustness, and stability of LiD-FL.

As a novel federated learning scheme, LiD-FL has great potential for future study. Promising directions for further research include the theoretical guarantee of LiD-FL for non-convex optimization tasks, extending LiD-FL to heterogeneous local data settings, and developing appropriate aggregation rules for LiD-FL.

Acknowledgments

References

Allen-Zhu, Z.; Ebrahimianghazani, F.; Li, J.; and Alistarh, D. 2021. Byzantine-Resilient Non-Convex Stochastic Gradient Descent. In *International Conference on Learning Representations (ICLR'21)*. OpenReview.net.

Allouah, Y.; Guerraoui, R.; Gupta, N.; Jellouli, A.; Rizk, G.; and Stephan, J. 2024a. Boosting Robustness by Clipping Gradients in Distributed Learning. arxiv:2405.14432.

Allouah, Y.; Guerraoui, R.; Gupta, N.; Pinot, R.; and Rizk, G. 2024b. Robust Distributed Learning: Tight Error Bounds and Breakdown Point Under Data Heterogeneity. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. Red Hook, NY, USA: Curran Associates Inc.

Allouah, Y.; Guerraoui, R.; Gupta, N.; Pinot, R.; and Stephan, J. 2023. On the Privacy-Robustness-Utility Trilemma in Distributed Learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*, volume 202 of *Proceedings of Machine Learning Research*, 569–626. PMLR.

Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; and Shmatikov, V. 2020. How To Backdoor Federated Learning. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS'20)*, volume 108 of *Proceedings of Machine Learning Research*, 2938–2948. PMLR.

Balcan, M.-F.; Blum, A.; and Vempala, S. 2008. A Discriminative Framework for Clustering via Similarity Functions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC'08)*, 671–680. New York, NY, USA: ACM.

Baruch, M.; Baruch, G.; and Goldberg, Y. 2019. A Little is Enough: Circumventing Defenses for Distributed Learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS'19)*. Red Hook, NY, USA: Curran Associates Inc.

Blanchard, P.; El Mhamdi, E. M.; Guerraoui, R.; and Stainer, J. 2017. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*, 118–128. Red Hook, NY, USA: Curran Associates Inc.

Bonawitz, K. A.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečný, J.; Mazzocchi, S.; McMahan, B.; Overveldt, T. V.; Petrou, D.; Ramage, D.; and Roselander, J. 2019. Towards Federated Learning at Scale: System Design. In *Proceedings of the Second Conference on Machine Learning and Systems (SysML'19)*, volume 1, 374–388. mlsys.org.

Caldas, S.; Duddu, S. M. K.; Wu, P.; Li, T.; Konečný, J.; McMahan, H. B.; Smith, V.; and Talwalkar, A. 2019. LEAF: A Benchmark for Federated Settings. arXiv:1812.01097.

Cao, X.; Fang, M.; Liu, J.; and Gong, N. Z. 2021. FLTrust: Byzantine-Robust Federated Learning via Trust Bootstrapping. In *28th Annual Network and Distributed System Security Symposium (NDSS'21)*. Virtual: Internet Society.

Charikar, M.; Steinhardt, J.; and Valiant, G. 2017. Learning from Untrusted Data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC'17)*, 47–60. ACM.

Cohen, G.; Afshar, S.; Tapson, J.; and van Schaik, A. 2017. EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN'17)*, 2921–2926. IEEE.

Diakonikolas, I.; Kane, D. M.; and Stewart, A. 2018. List-Decodable Robust Mean Estimation and Learning Mixtures of Spherical Gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC'18)*, 1047–1060. New York, NY, USA: ACM.

El-Mhamdi, E.-M.; Farhadkhani, S.; Guerraoui, R.; and Hoang, L.-N. 2023. On the Strategyproofness of the Geometric Median. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS'23)*, volume 206 of *Proceedings of Machine Learning Research*, 2603–2640. PMLR.

El Mhamdi, E. M.; Guerraoui, R.; and Rouault, S. 2018. The Hidden Vulnerability of Distributed Learning in Byzantium. In *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*, volume 80 of *Proceedings of Machine Learning Research*, 3521–3530. PMLR.

Fang, M.; Cao, X.; Jia, J.; and Gong, N. Z. 2020. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *Proceedings of the 29th USENIX Conference on Security Symposium (SEC'20)*, 1623–1640. USA: USENIX Association.

Farhadkhani, S.; Guerraoui, R.; Gupta, N.; and Pinot, R. 2024. On the Relevance of Byzantine Robust Optimization Against Data Poisoning. arxiv:2405.00491.

Farhadkhani, S.; Guerraoui, R.; Gupta, N.; Pinot, R.; and Stephan, J. 2022. Byzantine Machine Learning Made Easy By Resilient Averaging of Momentums. In *International Conference on Machine Learning (ICML'22)*, volume 162 of *Proceedings of Machine Learning Research*, 6246–6283. PMLR.

Gupta, N.; and Vaidya, N. H. 2020. Fault-Tolerance in Distributed Optimization: The Case of Redundancy. In *ACM Symposium on Principles of Distributed Computing (PODC'20)*, 365–374. ACM.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; D'Oliveira, R. G. L.; Eichner, H.; Rouayheb, S. E.; Evans, D.; Gardner, J.; Garrett, Z.; Gascón, A.; Ghazi,

B.; Gibbons, P. B.; Gruteser, M.; Harchaoui, Z.; He, C.; He, L.; Huo, Z.; Hutchinson, B.; Hsu, J.; Jaggi, M.; Javidi, T.; Joshi, G.; Khodak, M.; Konecný, J.; Korolova, A.; Koushanfar, F.; Koyejo, S.; Lepoint, T.; Liu, Y.; Mittal, P.; Mohri, M.; Nock, R.; Özgür, A.; Pagh, R.; Qi, H.; Ramage, D.; Raskar, R.; Raykova, M.; Song, D.; Song, W.; Stich, S. U.; Sun, Z.; Suresh, A. T.; Tramèr, F.; Vepakomma, P.; Wang, J.; Xiong, L.; Xu, Z.; Yang, Q.; Yu, F. X.; Yu, H.; and Zhao, S. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*, 14(1-2): 1–210.

Karimireddy, S. P.; He, L.; and Jaggi, M. 2021. Learning from History for Byzantine Robust Optimization. In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*, volume 139 of *Proceedings of Machine Learning Research*, 5311–5319. PMLR.

Karimireddy, S. P.; He, L.; and Jaggi, M. 2022. Byzantine-Robust Learning on Heterogeneous Datasets via Bucketing. In *10th International Conference on Learning Representations (ICLR'2022)*. OpenReview.net.

Karmalkar, S.; Klivans, A. R.; and Kothari, P. K. 2019. List-Decodeable Linear Regression. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS'19)*. Red Hook, NY, USA: Curran Associates Inc.

Konečný, J.; McMahan, H. B.; Ramage, D.; and Richtárik, P. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. arxiv:1610.02527.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images.

Lamport, L.; Shostak, R. E.; and Pease, M. C. 2019. The Byzantine Generals Problem. In Malkhi, D., ed., *Concurrency: the Works of Leslie Lamport*, 203–226. ACM.

Liu, S.; Gupta, N.; and Vaidya, N. H. 2021. Approximate Byzantine Fault-Tolerance in Distributed Optimization. In *ACM Symposium on Principles of Distributed Computing (PODC'21)*, 379–389. ACM.

Luan, Z.; Li, W.; Liu, M.; and Chen, B. 2024. Robust Federated Learning: Maximum Correntropy Aggregation Against Byzantine Attacks. *IEEE Transactions on Neural Networks and Learning Systems*, 1–14.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS'2017)*, volume 54 of *Proceedings of Machine Learning Research*, 1273–1282. PMLR.

Nguyen, D. C.; Pham, Q.; Pathirana, P. N.; Ding, M.; Seneviratne, A.; Lin, Z.; Dobre, O. A.; and Hwang, W. 2023. Federated Learning for Smart Healthcare: A Survey. *ACM Computing Surveys*, 55(3): 60:1–60:37.

Pillutla, K.; Kakade, S. M.; and Harchaoui, Z. 2022a. Robust Aggregation for Federated Learning. *IEEE Transactions on Signal Processing*, 70: 1142–1154.

Pillutla, K.; Kakade, S. M.; and Harchaoui, Z. 2022b. Robust Aggregation for Federated Learning. *IEEE Transactions on Signal Processing*, 70: 1142–1154.

Shejwalkar, V.; and Houmansadr, A. 2021. Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning. In *Proceedings of the 28th Network and Distributed System Security Symposium (NDSS'2021)*. The Internet Society.

Shejwalkar, V.; Houmansadr, A.; Kairouz, P.; and Ramage, D. 2022. Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning. In *Proceedings of IEEE Symposium on Security and Privacy (SP'22)*, 1354–1371. IEEE.

Sun, Z.; Kairouz, P.; Suresh, A. T.; and McMahan, H. B. 2019. Can You Really Backdoor Federated Learning? arxiv:1911.07963.

Xie, C.; Koyejo, O.; and Gupta, I. 2020. Fall of Empires: Breaking Byzantine-tolerant SGD by Inner Product Manipulation. In *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference (UAI'20)*, volume 115 of *Proceedings of Machine Learning Research*, 261–270. PMLR.

Yang, Q.; Liu, Y.; Chen, T.; and Tong, Y. 2019. Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2): 12:1–12:19.

Yi, Y.; You, R.; Liu, H.; Liu, C.; Wang, Y.; and Lv, J. 2024. Near-Optimal Resilient Aggregation Rules for Distributed Learning Using 1-Center and 1-Mean Clustering with Outliers. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'24)*, 16469–16477. AAAI Press.

Yin, D.; Chen, Y.; Ramchandran, K.; and Bartlett, P. L. 2018. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *Proceedings of the 35th International Conference on Machine Learning (ICML'2018)*, volume 80 of *Proceedings of Machine Learning Research*, 5636–5645. PMLR.

Zhao, P.; and Wan, Z. 2024. High Dimensional Distributed Gradient Descent with Arbitrary Number of Byzantine Attackers. arXiv:2307.13352.