ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240702

Tackling Selfish Clients in Federated Learning

Andrea Augello[a,*], Ashish Gupta[c], Giuseppe Lo Re[a] and Sajal K. Das[b]

[a]Department of Engineering, University of Palermo, Palermo, Italy
[b]Department of Computer Science, Missouri University of Science and Technology, Rolla, USA
[c]Department of Computer Science, BITS-Pilani, Dubai, UAE
Email: andrea.augello01@unipa.it, ashish@dubai.bits-pilani.ac.in, giuseppe.lore@unipa.it, sdas@mst.edu

Abstract. Federated Learning (FL) is a distributed machine learning paradigm facilitating participants to collaboratively train a model without revealing their local data. However, when FL is deployed into the wild, some intelligent clients can deliberately deviate from the standard training process to make the global model inclined toward their local model, thereby prioritizing their local data distribution. We refer to this novel category of misbehaving clients as selfish. In this paper, we propose a **R**obust aggregation strategy for the **FL** server to mitigate the effect of **Self**ishness (in short RFL-Self). RFL-Self incorporates an innovative method to recover (or estimate) the true updates of selfish clients from the received ones, leveraging robust statistics (median of norms) of the updates at every round. By including the recovered updates in aggregation, our strategy offers strong robustness against selfishness. Our experimental results, obtained on MNIST and CIFAR-10 datasets, demonstrate that just 2% of clients behaving selfishly can decrease the accuracy by up to 36%, and RFL-Self can mitigate that effect without degrading the global model performance.

1 Introduction

With an aim to train a Machine Learning (ML) model in a privacy-preserving manner, the researchers in [26] introduced Federated Learning (FL) framework, and since then it has drawn interest from the ML community. Being a distributed learning paradigm, FL enables each participant (or client) to train the model locally and send only model weights (parameters) to a central server for aggregation, thereby enabling learning from distributed heterogeneous devices without sharing their sensitive data. FL has been adopted in many contexts, such as medical records management [30], activity recognition [10], and smart homes [17]. Despite the advantages of FL, the lack of oversight over the training process can have serious implications. For instance, a client may deviate from the normal training [23], negatively affecting the underlying model. The deviation may be caused by a malicious client who wants to disrupt training [19], or by a normal client who has insufficient resources [16, 29]. Malicious clients may introduce noise in the model to prevent convergence, as in byzantine attacks, or optimize for a secondary objective, e.g., a backdoor [33]. As the server lacks control over the clients, preventing their harmful actions can be challenging.

The issue of malicious or adversarial clients is a very active field of FL. Broadly, two common approaches exist: (i) detect and remove the clients deviating from normal behavior [23, 36] and (ii) mitigate the impact of the misbehaving clients via robust aggregation [5, 25].

* Corresponding Author. Email: andrea.augello01@unipa.it.

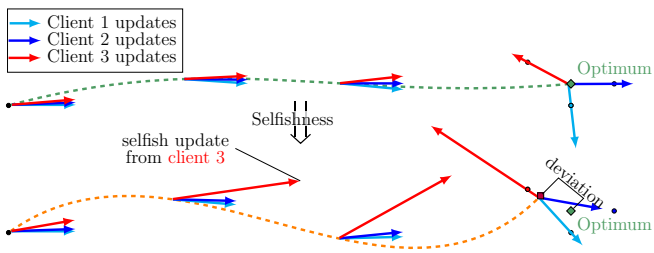


Figure 1: An example to show how selfish clients try to alter their updates to steer the global model toward their local optima.

For instance, the server in [14] excluded the contribution of colluding malicious clients who send similar updates. Robust aggregation strategies, employing statistics such as median and trimmed estimators, also exhibit good performance in the presence of byzantine clients without needing to identify them [12]. As long as the percentage of malicious clients stays below a certain threshold, the defense mechanism in [9] can theoretically provide the correct classification.

In this paper, we focus on a previously unexplored type of misbehaving client, the *selfish client*. Unlike malicious clients who intend to compromise the model, a selfish client is interested in making the global model prioritize its local data distribution. The resultant model may cause performance degradation for normal clients, especially in non-Independent and Identically Distributed (IID) settings, though it is not the goal of the selfish client. Figure 1 depicts an example of how selfishness deviates the global model from reaching the optimum. Typically, each participating client sends updates toward its local optimum, which the FL server aggregates to obtain a global optimum. However, a client behaving selfishly (shown by the 'red' arrows) sends updates that differ in magnitude and direction from the ones it usually sends.

Selfish updates cannot be directly included in the aggregation as that would lead to a deviated global model. However, unlike malicious clients, selfish clients do not have any malign intent. Their updates cannot be completely discarded as doing so would disregard their valuable contributions. This also discourages the adoption of prior defense methods (against malicious clients) [19, 16, 33] to deal with the selfishness issue. To the best of our knowledge, we are the first to tackle selfish clients in FL. We make the following major contributions:

- We introduce a novel concept of selfishness, caused by so-called selfish clients, in FL. Under non-IID settings, the presence of selfish clients is unavoidable as some intelligent clients can notice the difference between global and local optimum, and can try to exploit it in their favor.

- We propose an innovative **R**obust aggregation strategy that enables the **FL** server to mitigate the **Self**ishness effect from the training process (RFL-Self) without affecting the overall performance of the global model. In RFL-Self, the server aims to recover the true updates from the received selfish updates before aggregation.
- Through extensive experiments on two widely used datasets for FL, MNIST and CIFAR-10, we analyze the detrimental impact of selfish clients and demonstrate the effectiveness of RFL-Self for varying levels of selfishness and the number of selfish clients.

The paper is organized as follows. We first describe our selfish client behavior model and objective. Then, we introduce a selfish behavior implementation, assess the impact of selfish clients on FL, and present a mitigation method. We then discuss experimental results and finally conclude the paper.

2 Preliminaries and Problem Description

In FL, the goal of the server is to find an optimal global model $\mathbf{w}^*$ satisfying the optimization problem:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left\{ F(\mathbf{w}) \triangleq \frac{1}{k} \sum_{i \in [k]} F_i(\mathbf{w}) \right\}, \tag{1}$$

where $F_i(\mathbf{w})$ is the loss function of i-th client and $[k]$ is the set of all clients. To solve this optimization problem, at each communication round t, each client i computes a local update δ_i^t through Stochastic Gradient Descent (SGD). The updates are then sent to the server which computes a global update $\delta_{[k]}^t = \frac{1}{k} \sum_{i \in [k]} \delta_i^t$ as the average of all the local updates and obtains a new global model as $\mathbf{w}^{t+1} = \mathbf{w}^t + \delta_{[k]}^t$ for next round $t+1$. To avoid overburdening the notation, from now on we omit the round superscript when clear from context.

2.1 Selfish Client

The optimization objective in Eq. (1) aims to learn a globally optimal model rather than providing an optimal model for each client. In non-IID settings, the local optima of each client might differ from the global one. Some intelligent clients might notice this difference and be interested in obtaining a global model closer to their local optima. We refer to these savvy clients as *selfish* in FL. Through their actions, the global model ceases to be optimal for the whole system.

It is worth noting that using a personalized model for each client [32] would ensure that each client obtains a locally optimal, for instance by training multiple models for different groups of similar clients [2]. However, since selfish clients try to influence the global model at inference time, personalization techniques are unsuitable for their goals, as local fine-tuning has no impact on the performance of the global model. Thus, in this work, we focus on the case where a single global model is trained for all clients, which is the most common FL setup.

Selfish versus malicious clients: The notion of selfish clients completely differs from the malicious ones (adversaries) that are well defined in the literature [8]. Selfish behavior does deviate from normal behavior, but the objective is not explicitly in contrast with the global objective. Selfish clients lack nefarious intent, such as engaging in model poisoning attacks or preventing model convergence, as malicious clients do. Unlike backdoor attackers [33], selfish clients do not optimize the model for an additional objective different from the main task. The term "selfish" has been used in the context of FL in a complementary way in [24], where it refers to a server that strives to favor a subset of clients over others, which is the opposite problem to the one addressed in this work. Additionally, in certain works, "selfish" denotes typical malicious clients [35].

Further, in communication networks, selfish clients are those wishing to reap benefits from a system without contributing [28]. In FL, these clients are commonly defined as "free-riders" [13] who wish to obtain a model without engaging in the training process, exhibiting a completely different behavior than the one addressed in this work. A subset of researchers has utilized the term "selfish" to denote free-riders in their work [1, 31]. However, the problem tackled in these studies is unrelated to the one addressed in this paper. As far as we are aware, this study is the first to address a similar issue.

2.2 Problem Description

A selfish client $s \in [k]$ mainly aims to craft local updates such that, after aggregating the local updates from all the clients, the global model incurs a lower loss on its local data. The client s can formulate the objective for their local update as

$$\hat{\delta}_s = \arg\min_{\delta} \left\{ F_s \left(\hat{\mathbf{w}} \triangleq \mathbf{w} + \frac{\delta_{[k]\setminus\{s\}} + \delta}{k} \right) \right\}, \tag{2}$$

where the input to the objective function $F_s(\cdot)$ is not the global weights $\mathbf{w}$ but the crafted weights $\hat{\mathbf{w}}$ and two additional additive terms. The first term $\frac{\delta_{[k]\setminus\{s\}}}{k}$ involves knowing the sum of all the local updates of the rest of the clients, which is generally not known. The first term $\frac{\delta}{k}$ can be easily computed for any given k, but determining the optimal δ is not straightforward.

Though the selfish clients do not intentionally aim to pollute the training process, their actions are not innocent. When clients have non-IID data, the updates from selfish clients can considerably diverge the global model from the global optimum, causing performance degradation for normal clients.

To this end, the problem at hand is two-fold:

1. *From selfish client perspective:* solving the optimization problem in Eq. (2) given the global model weights $\mathbf{w}$ and k. Similar to [6], we assume clients know the total number of FL participants (k).
2. *From server perspective:* Alleviating the effect of selfishness on the global model through *robust aggregation*. Since selfish behavior differs from existing works on outliers and malicious clients, separate analysis and countermeasures are warranted.

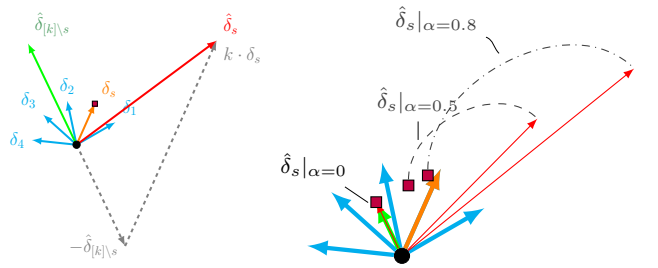
Due to the non-poisonous intent of the selfish clients, we can neither totally omit their updates from the aggregation nor include them directly, which makes the problem interesting and challenging compared to dealing with malicious clients.

3 Proposed Approach

To introduce selfish clients in the FL context, we first propose a novel way to estimate selfish model updates and then present a robust aggregation method to mitigate the effect of selfishness on the global model. This work considers two types of clients: normal and selfish. Each client has heterogeneous data and selfish clients do not collude.

3.1 Client Side: Computing Selfish Updates

To improve accuracy on its local data distribution, a selfish client needs to reduce the distance between global model updates $\delta_{[k]}$ (see



(a) (b)

Figure 2: (a) A visualization for how a selfish client can estimate $\hat{\delta}_s$ through model replacement. (b) Effect of α on the aggregated model and selfish update vector of a single selfish client. The aggregated model $\hat{\mathbf{w}}$ is obtained when $\hat{\delta}_s$ is included in the aggregation.

Eq. (2)) and the local model updates δ_s trained on local data. To do so, at each round, the selfish client estimates $\hat{\delta}_s$ using the received global model $\mathbf{w}$ and prior knowledge of k. It is worth noting that even if k is not known, the selfish client can still easily craft its updates approximating k, e.g., by jointly estimating it with $\delta_{[k]\setminus\{s\}}$. This alternative is further discussed in the Appendix [3] (Algorithm 2).

We propose an innovative strategy for the selfish client to estimate $\hat{\delta}_s$ to decrease the distance between the aggregated update $\delta_{[k]}$ and δ_s. The idea behind our strategy is partly inspired by the model replacement attacks [7], in which a malicious client attempts to replace the global model with its local model to neutralize the effect of other participants. The selfish client sends $\hat{\delta}_s$ in place of its true update δ_s, however, unlike model replacement attacks, it does not have malicious intent. A key distinction between model replacement attacks and selfish updates lies in their objectives. Model replacement attacks aim to manipulate the model towards a backdoor or impede convergence. Conversely, selfish updates seek to enhance the model performance based on (at least) some clients' data. In a perfect IID setting, we expect selfish updates to coincide with the true update, making the model converge to the same correct solution, unlike model replacement attacks. Part (a) of Figure 2 depicts how the selfish client's update $\hat{\delta}_s$ can deviate from the updates sent by normal clients. By sending $\hat{\delta}_s$, the training advances toward a global model closer to the selfish client's local optimum.

Challenge: To be effective, the selfish client needs to know the update $\delta_{[k]\setminus\{s\}}$ resulting from the sum of the other clients' updates. Obtaining this knowledge is generally not possible in a standard FL framework due to privacy concerns. Since we assume that the server is trustworthy and it communicates with clients via a secure channel, selfish clients cannot acquire $\hat{\delta}_{[k]\setminus\{s\}}$, thereby making it challenging to solve Eq. (2)

To deal with the above challenge, a selfish client can instead estimate the average update $\bar{\delta}_{[k]\setminus\{s\}}$ assuming that it does not change much between two consecutive communication rounds. Under this assumption, at round t, the selfish client uses the history of the global updates to compute an estimate of the average update $\bar{\delta}_{[k]\setminus\{s\}}$ as

$$\bar{\delta}^t_{[k]\setminus\{s\}} = \frac{k \cdot (\mathbf{w}^t - \mathbf{w}^{t-1}) - \hat{\delta}^{t-1}_s}{k-1} = \frac{\delta^{t-1}_{[k]\setminus\{s\}}}{k-1}, \tag{3}$$

where for $t = 1$, $\hat{\delta}^0_s = \delta^0_s$. This formulation is valid for neural network-based models, tree-based [22] models were not considered in this work. If multiple selfish clients are present, we assume that they are independent and each of them estimates their updates independently. After obtaining this estimate, the selfish client aims to bring the subsequent global update closer to its local one. To achieve this, the client might send $\hat{\delta}_s = k \cdot \delta_s - (k-1) \cdot \bar{\delta}_{[k]\setminus\{s\}}$ as an update to the server, completely replacing the global model, unintentionally resulting in malicious behavior. If the selfish client were to pursue this strategy, however, it would have no benefit in participating in the FL system. Thus, we introduce a selfishness parameter $\alpha \in [0, 1]$ to control the fraction of the global update that is replaced with $\hat{\delta}_s$ in every round. With a constant α across all rounds, the selfish client estimates its updates as

$$\begin{aligned}\hat{\delta}_s &= \alpha \left(k\delta_s - (k-1)\bar{\delta}_{[k]\setminus\{s\}}\right) + (1-\alpha)\bar{\delta}_{[k]\setminus\{s\}} \\ &= \alpha k(\delta_s - \bar{\delta}_{[k]\setminus\{s\}}) + \bar{\delta}_{[k]\setminus\{s\}}.\end{aligned} \tag{4}$$

When multiple selfish clients are present, we assume that they are independent and do not collude with each other. Each selfish client tries to optimize Eq. (2) independently using Eq. (4), irrespective of the presence of other selfish clients. Thus, the formulation holds with an arbitrary number of selfish clients, and we expect them to compete to increase their effect on the global model and try to cancel each other's effect.

A trivial alternative strategy that could be employed by a selfish client is to simply upscale its update. This strategy is equivalent to the presented one if $\delta_s = -\bar{\delta}_{[k]\setminus\{s\}}$, which is a special case of selfish update. In an adversarial setting, where the misbehaving client aims to harm the model convergence, upscaling can be equally effective, but the same is not true in our setting. In more realistic circumstances, upscaling can steer the model in a suboptimal direction.

Effect of parameter α: With $\alpha = 0$, the estimated update is equivalent to the other clients' updates, whereas, with $\alpha = 1$, the selfish client would overwrite the whole global update with its local update. In less extreme cases, selfishness can be regarded as the client trying to increase its effect during the aggregation process. If $\alpha = \frac{1}{k}$, then the update will simply be $\hat{\delta}_s|_{\alpha=\frac{1}{k}} = \delta_s$. On the other hand, $\hat{\delta}_s|_{\alpha=\frac{k-1}{k}} = (k-1)\delta_s - (k-2)\bar{\delta}_{[k]\setminus\{s\}}$, which affects the global model update as if all the clients but one have an update equivalent to the selfish client and the remaining client has the update equivalent to the average update of the normal clients. More generally, each $\frac{1}{k}$ increment of α increases the magnitude of the selfishness effect as if one more normal update δ_s from the selfish client were added and decreases the aggregated effect by one normal client. A visual representation of the effect of the parameter α on the selfish update vector and the aggregated model $\hat{\mathbf{w}}$ is shown in part (b) of Figure 2.

Impact of selfish updates: We perform initial experiments using the CIFAR-10 dataset [20] to demonstrate the impact of non-colluding selfish client(s) on the test accuracy of the global model with varying α. The results for two different FL setups (with $k = 50$) are reported in Figure 3. The clients' data distributions are non-IID with class partitioning so that each client has only the data of two randomly selected classes and every class of the dataset is assigned to at least one client. The global model is tested on each client's test set to evaluate its performance on their data distribution. More details on the experimental setup are given in Section 4. It is easy to notice that increasing the selfishness (i.e., increasing α) improves the accuracy for the selfish clients if there is only one in the FL setup. However, if a second selfish client appears, this is only true up to $\alpha = 0.3$. In the presence of two selfish clients with any same α, each of them tries to cancel out the other's updates, making the global updates unbounded in magnitude, and consequently, harming the convergence of the training process, which can be observed in part (b) of Figure 3 for $\alpha \geq 0.4$. In both cases, the performance of normal clients is harmed by the selfish clients. With 3 or more selfish clients without a mitigation strategy convergence is not achieved.

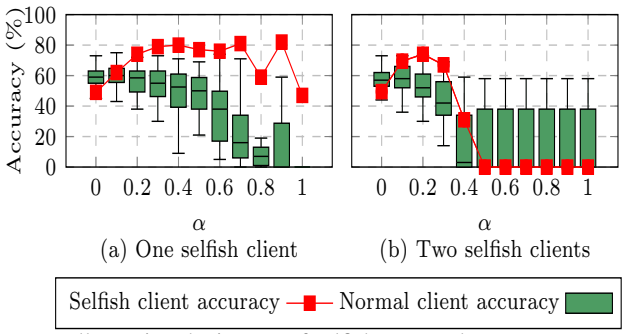


Figure 3: Illustrating the impact of selfishness on the test accuracy of the global model using CIFAR-10 dataset.

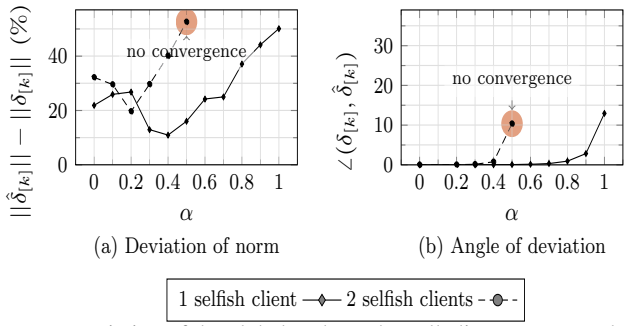


Figure 4: Deviation of the global update when all clients are normal to the one with selfish clients (CIFAR-10 dataset).

3.2 *Server Side: RFL-Self*

When selfish clients participate in FL, the objective of the server is to identify them before aggregation to mitigate their impact on the global model. However, even if these clients are detected correctly, they should not be completely excluded from the aggregation. Doing so leads to a model that performs poorly for the users (not available during training) having data distribution similar to the selfish clients. This can be observed in Figure 3 where with $\alpha = 0$ (no selfish client contribution), the performance of the global model decreases for the selfish client's data distribution. The only viable solution for the selfishness issue is estimating the true updates of selfish clients and using them in aggregation instead of the ones received from selfish clients. This section proposes a robust aggregation strategy, RFL-Self, to remove the effect of selfishness from the training process. In each round, RFL-Self identifies suspected selfish clients and attempts to recover their true update.

3.2.1 *Identifying Selfish Clients*

Given a set of received updates from all the clients, RFL-Self first computes the L_2 norm of each update and finds a median norm $\mathcal{N}_{med}$. Since a selfish client aims to increase its influence on the global model to deviate the training toward its local model, it is obvious that selfish updates are larger in magnitude than those from other clients. This intuition, confirmed by Theorem 1, suggests that for any client i, if $\|\delta_i\|^2 > \mathcal{N}_{med}$ holds, then client i might be selfish. In this case, the received update δ_i would be the crafted $\hat{\delta}_i$. This detection strategy is intentionally crafted to be simple and lightweight, but it is fully supported by the theoretical analysis in Theorem 1, and the experimental results confirm its effectiveness.

Figure 4 visually demonstrates the deviation of the global update $\delta_{[k]}$ caused by selfish updates. It showcases the disparity in norm and angle between the global update in scenarios where all clients are normal and where selfish clients are also present. It can be observed that part (a) of Figure 4 shows a U-shaped dip. This phenomenon is caused by the magnitude of $\hat{\delta}_s + \delta_{[k]\setminus\{s\}}$ not being equal to the magnitude of $\delta_{[k]}$, as also illustrated in part (b) of Figure 2. Even a single selfish client can severely affect global convergence, and just one more selfish participant may lead to a *no convergence* situation. Thus, recovering the true updates of the selfish clients becomes crucial.

3.2.2 *Recovering True Updates of Selfish Clients*

Upon successful identification of the selfish clients, there are three possible ways in which the server can deal with them:

- *Drop*: exclude suspicious updates from the aggregation.
- *Mitigate*: reduce the impact of suspicious updates.
- *Recover*: try and recover the genuine update.

The exclusion of selfish updates is not desirable as these updates are not malicious and help improve the generalization of the model. Moreover, normal clients might be wrongly identified as selfish in some rounds. Thus we focus on recovery to preserve valuable contributions. This approach is unique to our work as it does not apply to malicious clients.

RFL-Self uses the received selfish updates $\hat{\delta}_s$ to compute an estimate δ'_s of the true update δ_s and uses the estimate in the aggregation process. The server uses a convex combination of the received update $\hat{\delta}_s$ of each identified selfish client s and the marginal median update δ_{med} to obtain δ'_s as

$$\delta'_s = \beta\hat{\delta}_s + (1-\beta)\delta_{med}, \tag{5}$$

where the parameter β determines the convex combination and ensures that $||\delta'_s||$ is the same as the median of the norms of all the received updates $\mathcal{N}_{med}$. Thus, β can be computed by solving the equation

$$||\beta\hat{\delta}_s + (1-\beta)\delta_{med}|| = \mathcal{N}_{med}. \tag{6}$$

Through the convex combination expressed in Eq. (5), the potentially selfish update is scaled down and rotated toward the median update, obtaining a recovered version of the true update δ_s. Later in this section, we provide a theoretical analysis of the soundness of the recovery process and bounds to the possible reconstruction error.

3.2.3 *Robust Aggregation*

Finally, by replacing the received updates of the set of suspected selfish clients $\mathcal{S}$ with the recovered ones, the server aggregates the updates as expressed below:

$$\delta_{[k]} = \frac{1}{k}\left(\sum_{i\in[k]\setminus\mathcal{S}}\delta_i + \sum_{j\in\mathcal{S}}\delta'_j\right). \tag{7}$$

Later, the global model for the next round is computed as $\mathbf{w} = \mathbf{w} + \delta_{[k]}$. By substantially mitigating the effect of selfishness, our aggregation process offers a robust FL framework against the selfish participants while strategically utilizing their updates to achieve a more generalized global model.

Algorithm 1 summarizes all the steps of the RFL-Self. It is worth noting that RFL-Self differs from the standard aggregation method (FedAvg [26]) only at the server, thus it can be implemented as a drop-in replacement for FedAvg transparently from the clients without incurring any additional communication overhead.

Algorithm 1 RFL-Self algorithm

Input: A set of k clients
Output: A trained global model with weights $\mathbf{w}$

$\mathbf{w} \leftarrow$ random initialization
Send $\mathbf{w}$ to all the clients
for *each communication round* **do**
  Receive updates $\{\delta_i,\ \forall i \in [k]\}$
  Compute $\boldsymbol{\mathcal{N}} \leftarrow \{||\delta_i||,\ \forall i \in [k]\}$
  Compute a median norm $\mathcal{N}_{med}$ of $\boldsymbol{\mathcal{N}}$
  $\delta_{med} \leftarrow$ MARGINALMEDIAN$(\{\delta_i,\ \forall i \in [k]\})$
  $\mathcal{S} \leftarrow \{\}$
  for *each client* $i \in [k]$ **do**
    if $||\delta_i|| > \mathcal{N}_{med}$ **then**
      $\mathcal{S} \leftarrow$ APPEND(i)
  for *each client* $i \in \mathcal{S}$ **do**
    Obtain β by solving Eq. (6)
    Estimate δ'_i using Eq. (5)
  Update $\mathbf{w} \leftarrow \mathbf{w} + \frac{1}{k}(\sum_{i\in[k]\setminus\mathcal{S}} \delta_i + \sum_{j\in\mathcal{S}} \delta'_j)$
  Send the updated global model $\mathbf{w}$ to all the clients
return $\mathbf{w}$

On the server side, the overhead for computing the median of norms is $O(kd + k\log k)$, where d is the number of model parameters. This complexity is negligible compared to $O(dk\log k)$ required for the median computation. Thus, RFL-Self has asymptotically the same complexity as using the median as a robust aggregation mechanism and requires no additional communication.

3.2.4 *Theoretical Guarantees*

This section presents rigorous theoretical insights about the RFL-Self, focusing on the soundness of the detection and recovery mechanisms.

Theorem 1. *If the true update is similar in magnitude to the average update of the normal clients, then an effective estimated update of a selfish client is always larger in magnitude than the true update.*

Proof. Let us assume that there exists some α such that $\|\hat{\delta}_s\|^2 = \|\delta_s\|^2$, making the norm of the estimated update indistinguishable from the true update. Such values of α can be determined by using Eq. (4). In the norm space, this condition can be written as

$$\left\|\alpha k\delta_s + (1-\alpha k)\bar{\delta}_{[k]\setminus\{s\}}\right\|^2 = \|\delta_s\|^2,$$
$$(\alpha k\|\delta_s\|)^2 + 2\alpha k(1-\alpha k)\langle\delta_s, \bar{\delta}_{[k]\setminus\{s\}}\rangle + (1-\alpha k)^2\left\|\bar{\delta}_{[k]\setminus\{s\}}\right\|^2 = \|\delta_s\|^2,$$

where $\langle\cdot,\cdot\rangle$ is the inner product. After solving the above, we get

$$\alpha = \frac{1}{k} \quad \text{and} \quad \alpha = \frac{\left\|\bar{\delta}_{[k]\setminus\{s\}}\right\|^2 - \|\delta_s\|^2}{k(\|\delta_s\|^2 + 2\langle\delta_s, \bar{\delta}_{[k]\setminus\{s\}}\rangle + \|\bar{\delta}_{[k]\setminus\{s\}}\|^2)} \tag{8}$$

For $\alpha = \frac{1}{k}$ the $\hat{\delta}_s = \delta_s$, i.e., the estimated update coincides with the true update. The other value for α in Eq. (8), if $\|\delta_s\|^2 \simeq \|\bar{\delta}_{[k]\setminus\{s\}}\|^2$, corresponds to $\alpha \simeq 0$, i.e. $\hat{\delta}_s \simeq \bar{\delta}_{[k]\setminus\{s\}}$. Thus, the only way to avoid updates with a larger magnitude (larger norm) than the true ones, is by not behaving selfishly, while effective selfish updates are always larger in magnitude than the true updates. □

Remark. Theorem 1 does not ensure that all clients exhibiting larger update norms are necessarily acting selfishly. Consequently, RLF-Self can incur false positives. This aspect is taken into account in the recovery process, ensuring that the simplicity of the detection mechanism does not undermine performance.

Recovery error: The convex combination in Eq. (5) reduces the effect of the selfish update steering it towards the normal clients' updates. More formally, we analyze the recovery error of RFL-Self under the following conditions:

- *Condition 1:* The average update $\delta_{[k]\setminus\{s\}}$ (excluding selfish updates) is similar between any two consecutive rounds.
- *Condition 2:* The median update δ_{med} is a good estimator of the mean update of all but the selfish clients.
- *Condition 3:* The true update by a selfish client δ_s is close in magnitude to $\mathcal{N}_{med}$.

If the first two conditions hold, then $\bar{\delta}_{[k]\setminus\{s\}} \simeq \delta_{med}$ and from Eqs. (4) and (5) we get

$$\begin{aligned}\delta'_s &\simeq \beta[\alpha k(\delta_s - \bar{\delta}_{[k]\setminus\{s\}}) + \bar{\delta}_{[k]\setminus\{s\}}] + (1-\beta)\delta_{med} \\ &\simeq \beta\alpha k\delta_s + (1-\beta\alpha k)\bar{\delta}_{[k]\setminus\{s\}}.\end{aligned} \tag{9}$$

This recovered update δ'_s coincides with δ_s for $\beta = 1/(\alpha k)$. The server, however, does not know of α, hence, it chooses β by solving $||\delta'_s|| = \mathcal{N}_{med}$, as mentioned in Eq. (6). If all three conditions hold, then choosing β using Eq. (6) would provide $\beta \simeq 1/(\alpha k)$, thereby taking δ'_s quite close to δ_s. Part (a) of Figure 5 shows the effectiveness of RFL-Self on recovery of the selfish update δ'_s if the three conditions are met.

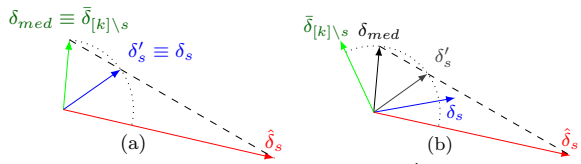


Figure 5: Distance of the recovered update δ'_s to δ_s: (a) when the conditions 1-3 are met, and (b) when the three conditions do not hold.

Even if all the conditions are not satisfied perfectly, the RFL-Self can still provide a good approximation. For example, dropping Condition 2, we get $\delta'_s = \beta\alpha k\delta_s + (\delta_{med} - \beta\alpha k\bar{\delta}_{[k]\setminus\{s\}}) + \beta(\bar{\delta}_{[k]\setminus\{s\}} - \delta_{med})$, which differs by $(1-\beta)(\delta_{med} - \bar{\delta}_{[k]\setminus\{s\}})$ from the one in Eq. (9). The norm of this difference (error) is upper bounded by $\sqrt{\mathrm{Tr}(\mathrm{var}(\delta))}$ [15]. Similarly, dropping Condition 3, the estimated value will be closer to the median update than the original one, with a maximum error of $||\delta_s - \delta_{med}||$. Part (b) of Figure 5 shows the difference between the recovered update δ'_s and the original update δ_s when the three conditions do not hold. It is worth mentioning that a trivial mitigation strategy, such as downscaling selfish updates by a factor β before aggregation, does not provide the same guarantees. Downscaling yields $\delta'_s = \frac{\hat{\delta}_s}{\beta} = \frac{\alpha k}{\beta}\delta_s + \frac{1-\alpha k}{\beta}\bar{\delta}_{[k]\setminus\{s\}}$. No matter how β is chosen, the estimated update δ'_s cannot match the true update. RFL-Self offers greater robustness by estimating δ'_s closer to the true update δ_s, as also empirically demonstrated in Section 4.

Theorem 2. *The maximum error in the recovered aggregated update is bounded by* $\frac{4+k}{4k}\,\mathrm{Tr}(\mathrm{var}(\delta))$.

Proof. Owing to Theorem 1, in the presence of a set $\mathcal{S}$ of suspected selfish clients, the expected error is:

$$\begin{aligned}\mathbb{E}\left\|\delta_{[k]} - \delta'_{[k]}\right\|^2 &= \tfrac{1}{k^2}\mathbb{E}\left\|\textstyle\sum_{i\in[k]}\delta_i - \left(\sum_{i\in[k]\setminus\mathcal{S}}\delta_i + \sum_{j\in\mathcal{S}}\delta'_j\right)\right\|^2 \\ &= \tfrac{1}{k^2}\mathbb{E}\left\|\textstyle\sum_{i\in\mathcal{S}}\delta_i - \sum_{j\in\mathcal{S}}\delta'_j\right\|^2\end{aligned} \tag{10}$$

Since the RFL-Self leaves all $[k] \setminus \mathcal{S}$ updates unchanged, the error comes from the updates whose norm is larger than $\mathcal{N}_{med}$. The maximum error induced by recovering these updates is obtained by using the median update instead

$$\frac{1}{k^2}\mathbb{E}\left\|\sum_{i\in\mathcal{S}}\delta_i-\sum_{j\in\mathcal{S}}\delta'_j\right\|^2\leq\frac{1}{k^2}\mathbb{E}\left\|\sum_{i\in\mathcal{S}}(\delta_i-\delta_{med})\right\|^2$$
$$=\frac{1}{k^2}\mathbb{E}\left\|\sum_{i\in\mathcal{S}}\left(\delta_i-\bar{\delta}_{[k]}-\delta_{med}+\bar{\delta}_{[k]}\right)\right\|^2$$

Using Jensen's inequality [18]

$$\leq\frac{1}{k^2}\mathbb{E}\left\|\sum_{i\in\mathcal{S}}\left(\delta_i-\bar{\delta}_{[k]}\right)\right\|^2+\frac{|\mathcal{S}|^2}{k^2}\mathrm{Tr}(\mathrm{var}(\delta))$$
$$\leq\frac{1}{k}\mathrm{Tr}(\mathrm{var}(\delta))+\frac{|\mathcal{S}|^2}{k^2}\mathrm{Tr}(\mathrm{var}(\delta)) \quad (11)$$
$$\simeq\frac{4+k}{4k}\mathrm{Tr}(\mathrm{var}(\delta)) \quad \text{since } |\mathcal{S}|\simeq\frac{k}{2}. \quad \square$$

Remark. According to Theorem 2, the error does not depend on the number of selfish clients as long as they are insufficient to induce substantial bias into the median.

4 Experimental Evaluation

This section empirically analyzes the impact of selfish clients on FL system performance and evaluates the effectiveness of our RFL-Self method. Under an image classification task, we use two benchmark datasets widely adopted to assess FL algorithms, MNIST [11] and CIFAR-10 [20], with 50 clients varying the level of selfishness α with up to 20% of the clients being selfish. We chose to assess RFL-Self on these datasets due to their widespread use as benchmarks for FL and their varying complexities. Larger datasets like CIFAR-100 were not included because, in non-IID settings, the increased diversity in data distribution compels selfish clients to transmit even larger updates. Consequently, detecting selfish clients using our method becomes straightforward. To simulate the non-IID scenario, we partition the dataset so that each client has data for two randomly selected classes, which is the most challenging setting [21], as few clients have overlapping classes. The tested non-IID conditions represent one of the most challenging scenarios in FL. In less severe non-IID conditions, the influence of selfish clients is expected to be less pronounced. Additional experiments on a smaller FL setup involving 5 clients are detailed in the Appendix [3]

Experimental setup: For the MNIST dataset, we train a CNN with 2 convolutional layers followed by 2 fully connected layers. For the CIFAR-10 dataset, the trained CNN has 3 convolutional layers instead. Our investigation of FL is focused solely on the perspective of selfishness, rather than maximizing accuracy. Therefore, these shallow models are suitable enough for our task. The hyper-parameters are – *optimizer*: SGD, *batch size*: 128/256, and *learning rate*: 0.01 and 0.1 for MNIST and CIFAR-10 datasets, respectively. We train the models for 30 communication rounds, with five local epochs per round at each client, assuming full participation of all the clients in every round. All the experiments are implemented in Python using a well-known library PyTorch 1.12.1 [27] on a Windows 11 powered with an NVIDIA RTX A5000 GPU. The source code for this work is available at [4].

4.1 Impact of Selfish Clients on Model Performance

At first, we carry out experiments to analyze the impact of selfish client(s) on the global model accuracy without using RFL-Self, and report results in Figure 6 for the MNIST dataset. Similar results were discussed in Figure 3 for CIFAR-10 to ease the understanding of the context therein. Though the mean test accuracy (shown by 'green' box plots) across the normal clients keeps decreasing as the selfishness level α increases, it does not seem to favor the selfish clients. In part (b) of Figure 6, for $\alpha > 0.3$ the two selfish clients also start losing accuracy. It is interesting to observe that in the presence of two selfish

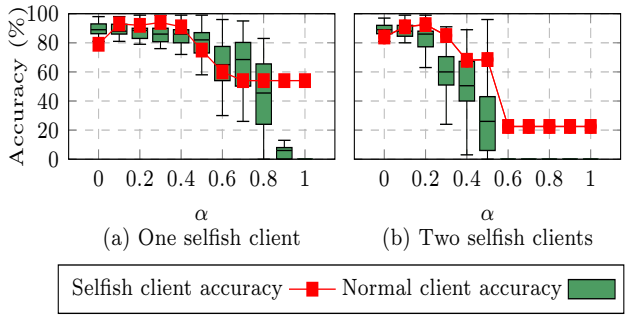


Figure 6: Test accuracy of the global model varying α with one and two selfish clients on the MNIST dataset.

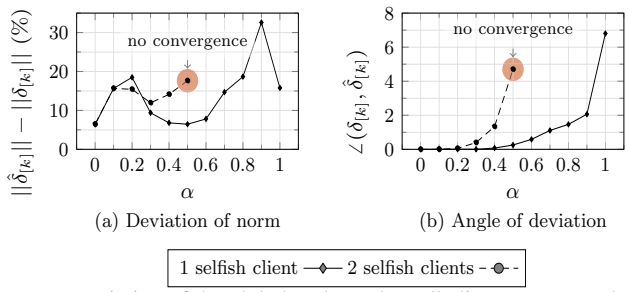


Figure 7: Deviation of the global update when all clients are normal to the one with selfish clients (MNIST dataset).

clients, none of them gets much benefit, rather each seems to cancel out the other's updates, thus getting trapped into a no-win situation. Additionally, the selfishness causes a higher variance of the test accuracy of the global model across normal clients, thus decreasing the overall performance of the FL system, and in turn, making the model more unpredictable and unfair to the normal participants.

In addition, to analyze how selfishness affects the model convergence, we compute the deviation between the global update in the absence of selfish clients $\delta_{[k]}$ and the global update in the presence of selfish client(s) without applying RFL-Self $\hat{\delta}_{[k]}$. The obtained results for the MNIST dataset are reported in Figure 7. We also showed similar results for CIFAR-10 in Figure 4 to establish the need for our aggregation method. An important point to notice here is that in the presence of two selfish clients, at $\alpha = 0.5$, the global model fails to converge though the deviation is not large. It is happening because each selfish client cancels out the selfishness component of the other, thereby the overall model weights keep fluctuating. These results without any mitigation strategy further emphasize the necessity of a robust aggregation mechanism to handle selfish clients in FL systems.

4.2 Performance of RFL-Self

Next, we evaluate the performance of RFL-Self in the presence of selfish clients and make a fair comparison with two standard strategies: median and downscaling. The median is also a robust aggregation strategy in which the median is used instead of the mean when aggregating, it is commonly used in FL systems to deal with outliers and byzantine clients [34], and is representative of the "drop" strategy that completely excludes the updates of the suspected selfish clients from the aggregation process. In the downscaling strategy, whenever a client is suspected to be selfish in a given round, instead of using Eq. (5), we scale each suspected update by $\mathcal{N}_{med}/\|\hat{\delta}_s\|$. The authors in [16] adopted downscaling to deal with unreliable clients, here we use it as a representative of the "mitigate" strategy. While sophisticated malicious client detection mechanisms have been proposed in the literature, in our scenario, Theorem 1 proves that our criterion,

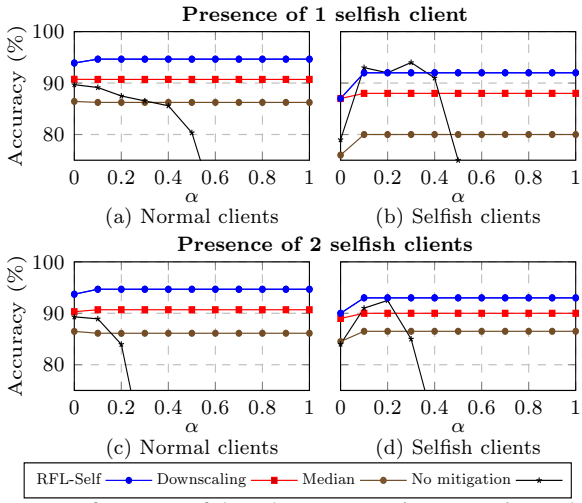


Figure 8: Performance of the robust aggregation strategies under an FL setup with 50 clients using MNIST dataset.

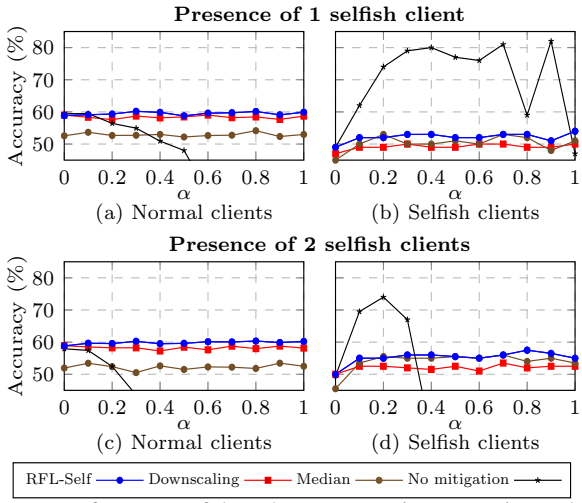


Figure 9: Performance of the robust aggregation strategies under an FL setup with 50 clients using CIFAR-10 dataset.

albeit simple (with a low computational overhead), is guaranteed to be effective. The main contribution on the defense side of the work lies in the original update recovery, not in the flagging mechanism. Nevertheless, the proposed flagging mechanism can be substituted with different ones without affecting the recovery process.

We test all three strategies in the same settings as the selfish client impact assessment. The test accuracy results for selfish and normal clients with varying α are reported in Figures 8 and 9.

A quick observation from the results is that selfishness with $\alpha > 0.4$ in FL can cause severe accuracy degradation for both normal and selfish clients if no robust aggregation has been used at the server. An ideal mitigation strategy should prevent accuracy reduction for normal clients without affecting selfish clients. In that sense, all the considered strategies seem to perform well by preventing selfish clients from causing drastic accuracy reductions. Additionally, the RFL-Self method outperforms the downscaling by $4\% \sim 5\%$ and the median by $7\% \sim 12\%$ on MNIST dataset for all $\alpha > 0$, because our method avoids excessive penalties for selfish clients while still preventing them from gaining much on the accuracy. Interestingly, selfish clients never achieved more accuracy than normal ones regardless of the selfishness level. Another significant observation is that, even when selfishness levels are low, RFL-Self outperforms the "No mitigation" scenario. Even in the absence of selfish clients, the proposed method does not compromise performance.

4.3 *Performance with More Selfish Clients*

Finally, we investigate how the performance of the considered strategies changes by increasing the number of selfish clients with $\alpha \in \{0.2, 0.3, 0.4\}$. We choose these values of α with the help of part (b) of Figure 3, where at $\alpha = 0.2$, the model achieves maximum accuracy even with two selfish clients and right after that it starts degrading and at $\alpha = 0.4$, we notice a steep drop in accuracy. Table 1 reports the results on the CIFAR-10 dataset under an FL setup with 50 clients including the selfish clients (0% to 20%). On the CIFAR-10 dataset, at $\alpha = 0.3$ with 10% selfish clients, the downscaling loses to our method by a substantial 4.2% margin $(60.84 - 56.64)$. Regardless of the percentage of selfish clients, RFL-Self outperforms both downscaling and median and maintains a fair balance between the accuracy of normal and selfish clients. Notably, RFL-Self outperforms the other strategies even in the absence of selfish clients, supporting that it does not harm the performance of normal clients.

Table 1: Performance of the global model and standard deviation under the different aggregation strategies on the CIFAR-10 dataset.

α	Selfish % (# clients)		Normal clients accuracy RFL-Self	Downscaling	Median	Selfish clients accuracy RFL-Self	Downscaling	Median
0.2	0	(0)	**59.58**±6.54	59.28±6.23	52.42±7.11	-	-	-
	5	(3)	**59.11**±6.17	57.83±6.31	57.67±9.18	**56.67**±1.35	55.00±1.94	54.53±3.57
	10	(5)	**57.78**±6.52	56.91±7.04	55.64±8.41	**57.80**±2.40	54.80±2.26	56.60±7.74
	20	(10)	**56.92**±7.11	55.95±8.72	52.12±9.64	**60.00**±6.13	58.30±8.58	55.00±8.26
0.3	0	(0)	**59.58**±6.54	59.28±6.23	52.42± 7.11	-	-	-
	5	(3)	**59.64**±6.24	58.13±6.65	53.98± 9.89	**55.33**±1.88	53.00±2.08	54.33± 4.73
	10	(5)	**60.84**±6.76	56.64±6.22	55.47±10.18	**56.00**±2.12	54.20±1.81	56.20± 8.05
	20	(10)	**56.52**±6.28	55.88±6.10	52.80± 9.20	**59.80**±6.88	58.20±6.96	56.20±10.19
0.4	0	(0)	**59.58**±6.54	59.28±6.23	52.42± 7.11	-	-	-
	5	(3)	**59.94**±6.16	58.51±6.13	53.15± 9.51	**55.67**±2.49	52.33±2.65	54.33±4.16
	10	(5)	**60.49**±6.49	58.91±8.73	54.33±10.38	**56.00**±2.23	54.40±6.37	49.60±7.33
	20	(10)	**60.08**±9.83	59.05±9.09	53.77± 9.01	**58.70**±9.08	58.30±9.23	54.80±8.79

Table 2 reports the obtained results for the MNIST dataset with 50 clients including the selfish clients (0% to 20%). On a simple dataset like MNIST, the normal clients can achieve more than 90% accuracy via downscaling, closer to the performance of RFL-Self.

Table 2: Performance of the global model and standard deviation under the different aggregation strategies on the MNIST dataset.

α	Selfish % (# clients)		Normal clients accuracy RFL-Self	Downscaling	Median	Selfish clients accuracy RFL-Self	Downscaling	Median
0.2	0	(0)	**90.98** ± 3.66	90.66 ± 3.82	86.22 ± 3.86	-	-	-
	5	(3)	**92.34** ± 3.55	91.81 ± 3.96	88.40 ± 3.83	**92.00** ± 2.11	91.33 ± 3.75	89.00 ± 2.49
	10	(5)	**92.67** ± 3.64	90.40 ± 3.90	91.78 ± 3.87	**93.20** ± 1.87	91.20 ± 1.71	92.20 ± 1.83
	20	(10)	**92.65** ± 3.74	91.28 ± 4.36	88.20 ± 4.15	**92.80** ± 3.38	91.80 ± 3.96	90.60 ± 4.01
0.3	0	(0)	**90.98** ± 3.66	90.66 ± 3.82	86.22 ± 3.86	-	-	-
	5	(3)	**90.85** ± 3.69	90.68 ± 3.82	86.13 ± 3.78	**91.00** ± 2.16	90.33 ± 3.06	86.33 ± 2.49
	10	(5)	**90.58** ± 3.73	90.40 ± 3.95	86.13 ± 4.00	**91.60** ± 1.85	91.20 ± 1.82	88.00 ± 1.96
	20	(10)	**90.70** ± 3.80	90.20 ± 4.07	86.05 ± 4.12	**91.30** ± 3.41	90.60 ± 3.80	88.40 ± 4.01
0.4	0	(0)	**90.98** ± 3.66	90.66 ± 3.82	86.22 ± 3.86	-	-	-
	5	(3)	**90.85** ± 3.70	90.66 ± 3.95	86.34 ± 3.82	**91.00** ± 2.16	90.67 ± 2.31	86.33 ± 2.39
	10	(5)	**90.58** ± 3.74	90.40 ± 4.00	86.13 ± 3.79	**91.60** ± 1.85	91.20 ± 1.79	88.00 ± 1.78
	20	(10)	**90.70** ± 3.81	90.20 ± 4.12	86.03 ± 4.06	**91.30** ± 3.40	90.60 ± 4.01	88.40 ± 4.02

5 Conclusion

We introduced a novel notion of *selfish clients* who can deviate the overall FL training in their favor. From the server perspective, we proposed a robust aggregation strategy, RFL-Self, to mitigate the impact of these clients on the global model. With rigorous analysis, we established that selfish clients can severely affect the training process and potentially deviate it to no convergence point. By recovering true updates of selfish clients, the RFL-Self offered a strong robust aggregation strategy against selfishness. By conducting extensive empirical analysis using two benchmark datasets with varying levels of selfishness, we observed that RFL-Self can handle the selfishness without degrading the model accuracy for normal clients, and it is superior to other standard strategies like downscaling and median. In the future, we plan to investigate adaptive selfishness, collusion among selfish clients, and the impact of selfish clients on fairness.

Acknowledgements

This work is supported by NSF grants under award numbers CNS-2008878, SaTC-2030624, and ECCS-2319995.

References

[1] S. Arisdakessian, O. A. Wahab, A. Mourad, and H. Otrok. Coalitional federated learning: Improving communication and training on non-iid data with selfish clients. *IEEE Trans. on Services Computing*, 2023.

[2] A. Augello, G. Falzone, and G. Lo Re. DCFL: Dynamic Clustered Federated Learning under Differential Privacy Settings. In *2023 IEEE Int. Conf. on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 614–619, 2023. doi: 10.1109/PerComWorkshops56833.2023.10150285.

[3] A. Augello, A. Gupta, G. Lo Re, and S. K. Das. Tackling selfish clients in federated learning. *arXiv preprint:2407.15402*, July 2024. URL https://arxiv.org/abs/2407.15402.

[4] A. Augello, A. Gupta, G. Lo Re, and S. K. Das. Rfl-self, July 2024. URL https://doi.org/10.5281/zenodo.12755207.

[5] S. Awan, B. Luo, and F. Li. Contra: Defending against poisoning attacks in federated learning. In *Proc. of the 26th European Symp. on Research in Computer Security*, pages 455–475. Springer, 2021.

[6] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. In *Proc. of the Int. Conf. on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[7] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo. Analyzing Federated Learning through an Adversarial Lens. In *Proc. of the 36th Int. Conf. on Machine Learning*, pages 634–643. PMLR, 2019. URL https://proceedings.mlr.press/v97/bhagoji19a.html.

[8] A. Blanco-Justicia, J. Domingo-Ferrer, S. Martínez, D. Sánchez, A. Flanagan, and K. E. Tan. Achieving security and privacy in federated learning systems: Survey, research challenges and future directions. *Engineering Applications of Artificial Intelligence*, 106:104468, 2021.

[9] X. Cao, J. Jia, and N. Z. Gong. Provably secure federated learning against malicious clients. In *Proc. of the AAAI conference on artificial intelligence*, pages 6885–6893, 2021.

[10] F. Concone, C. Ferdico, G. Lo Re, and M. Morana. A Federated Learning Approach for Distributed Human Activity Recognition. In *Proc. of the IEEE Int. Conf. on Smart Computing (SMARTCOMP)*, pages 269–274, 2022.

[11] L. Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[12] E. M. El Mhamdi, R. Guerraoui, and S. Rouault. The hidden vulnerability of distributed learning in Byzantium. In *Proc. of the 35th Int. Conf. on Machine Learning*, volume 80, pages 3521–3530. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/mhamdi18a.html.

[13] Y. Fraboni, R. Vidal, and M. Lorenzi. Free-rider Attacks on Model Aggregation in Federated learning. In *Proc. of The 24th Int. Conf. on Artificial Intelligence and Statistics*, volume 130, pages 1846–1854. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/fraboni21a.html.

[14] C. Fung, C. J. Yoon, and I. Beschastnikh. The Limitations of Federated Learning in Sybil Settings. In *RAID*, pages 301–316, 2020.

[15] R. Garver. Concerning the limits of a measure of skewness. *The Annals of Mathematical Statistics*, 3(4):358–360, 1932.

[16] A. Gupta, T. Luo, M. V. Ngo, and S. K. Das. Long-Short History of Gradients Is All You Need: Detecting Malicious and Unreliable Clients in Federated Learning. In *Proceeedings of the 27th European Symp. on Research in Computer Security*, pages 445–465, 2022. ISBN 978-3-031-17143-7.

[17] A. Gupta, H. P. Gupta, and S. K. Das. FedAR+: A Federated Learning Approach to Appliance Recognition with Mislabeled Data in Residential Environments. In *Proc. of the ACM/IEEE 14th Int. Conf. on Cyber-Physical Systems (with CPS-IoT Week 2023)*, pages 78–87, 2023.

[18] J. L. W. V. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30(1):175–193, 1906.

[19] M. S. Jere, T. Farnan, and F. Koushanfar. A taxonomy of attacks on federated learning. *IEEE Security & Privacy*, 19(2):20–28, 2021. doi: 10.1109/MSEC.2020.3039941.

[20] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[21] Q. Li, Y. Diao, Q. Chen, and B. He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th Int. Conf. on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022.

[22] Q. Li, W. ZHAOMIN, Y. Cai, y. han, C. M. Yung, T. Fu, and B. He. Fedtree: A federated learning system for trees. In D. Song, M. Carbin, and T. Chen, editors, *Proc. of Machine Learning and Systems*, volume 5, pages 89–103. Curan, 2023.

[23] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen. Abnormal client behavior detection in federated learning. In *Int. Workshop on Federated Learning for User Privacy and Data Confidentiality, in Conjunction with NeurIPS*, 2019.

[24] R. Luo, S. Hu, and L. Yu. Rethinking client reweighting for selfish federated learning. 2021.

[25] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng. ShieldFL: Mitigating model poisoning attacks in privacy-preserving federated learning. *IEEE Trans. on Information Forensics and Security*, 17:1639–1654, 2022.

[26] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2, 2016.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[28] A. Rahim, X. Kong, F. Xia, Z. Ning, N. Ullah, J. Wang, and S. K. Das. Vehicular social networks: A survey. *Pervasive and Mobile Computing*, 43:96–113, 2018.

[29] A. Reisizadeh, I. Tziotis, H. Hassani, A. Mokhtari, and R. Pedarsani. Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity. *IEEE Journal on Selected Areas in Information Theory*, 3(2):197–205, 2022.

[30] N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):119, 2020.

[31] Y. E. Sagduyu. Free-rider games for federated learning with selfish clients in nextg wireless networks. In *2022 IEEE Conf. on Communications and Network Security (CNS)*, pages 365–370. IEEE, 2022.

[32] A. Z. Tan, H. Yu, L. Cui, and Q. Yang. Towards Personalized Federated Learning. *IEEE Trans. on Neural Networks and Learning Systems*, pages 1–17, 2022. ISSN 2162-2388. doi: 10.1109/TNNLS.2022.3160699.

[33] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33:16070–16084, 2020.

[34] D. Yin, Y. Chen, R. Kannan, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proc. of the Int. Conf. on Machine Learning*, pages 5650–5659. PMLR, 2018.

[35] B. Youssef, D. Jabir, and K. Abdellatif. Impact of selfish nodes on federated learning performances. In *2022 9th Int. Conf. on Wireless Networks and Mobile Communications (WINCOM)*, pages 1–6. IEEE, 2022.

[36] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong. FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proc. of the 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, pages 2545–2555, 2022.