Enhancing Robustness of Federated Learning via Server Learning

Van Sy Mai[1], Richard J. La[2], Kushal Chakrabarti[3], and Dipankar Maity[4]

***Abstract*—This paper explores the use of server learning for enhancing the robustness of federated learning against malicious attacks even when clients' training data are not independent and identically distributed. We propose a heuristic algorithm that uses server learning and client update filtering in combination with geometric median aggregation. We demonstrate via experiments that this approach can achieve significant improvement in model accuracy even when the fraction of malicious clients is high, even more than 50% in some cases, and the dataset utilized by the server is small and could be synthetic with its distribution not necessarily close to that of the clients' aggregated data.**

I. Introduction

Federated Learning (FL) has emerged as a means of distributed learning using local data stored at clients with a coordinating server. Recent studies showed that FL can suffer from poor performance when training data at the clients are not independent and identically distributed (IID) and more severely so under Byzantine attacks, where malicious participants can disrupt training by injecting faulty updates.

In general, there are two main strategies to enhance the robustness of FL against Byzantine attacks. The first is to use robust aggregation techniques, e.g., coordinate-wise trimmed mean or geometric median, to mitigate the impact of malicious updates [1], [2]. This approach relies on a key assumption that the fraction of malicious clients is always less than a half and usually an upper bound on this fraction is available. The second approach is to use auxiliary information to filter out potentially malicious updates before the model aggregation step. This relies on quality auxiliary information, such as additional data available at the server [3].

Recently, [4] proposed an approach to improving FL on non-IID client data using server learning (SL) even when the server dataset is small and not necessarily close to client data distribution. Updates from malicious clients can be viewed as those from out-of-distribution and, hence, SL has the potential to improve the overall performance. A natural question is the following: *Can SL improve the robustness of FL against Byzantine attacks*? We propose to use both SL and filtering in combination with geometric median aggregation to improve the robustness of FL on non-IID data and under Byzantine attacks. Our experiments show that such combination can achieve significant improvements in model accuracy even when the fraction of malicious clients is high and the dataset utilized by the server is small and its distribution differs from that of the clients' aggregate data.

The proposed approach integrates two key methods to enhance the robustness of FL against Byzantine attacks, particularly in non-IID settings: the *use of a server-side dataset for learning and filtering*, and the application of *robust aggregation mechanisms*. This positions our work at the intersection of several active research areas in FL, as briefly outlined next.

Byzantine-Robust Aggregation Rules: A primary line of defense against Byzantine attacks involves developing robust aggregation rules at the server to mitigate the impact of malicious model updates. Early and prominent methods include Krum and Multi-Krum, which select a singe client update that is closest to its neighbors [1], and coordinate-wise median and trimmed mean, which are robust to outliers in individual dimensions of the model update vector. Our approach utilizes geometric median, a well-established robust aggregator that can tolerate up to half of the clients being malicious [2]. More recent work continued to refine these aggregation strategies. However, a key limitation of relying solely on aggregation is the assumption that malicious clients constitute a minority, a condition that may not hold in all scenarios. The proposed approach aims to overcome this by integrating SL.

Filtering Malicious Updates: Another strategy to defend against Byzantine attacks is to filter out malicious updates before the aggregation step. This can be achieved through various techniques, including anomaly detection and clustering-based approaches that identify and exclude suspicious updates [5]. Another approach in [6] aims to detect malicious clients based on their model-updates consistency, which could be ineffective against data poisoning, especially in non-IID settings. The study in [3] assumes that a server uses a small dataset with the same distribution as the global training data for filtering. Some methods leverage similarity metrics, such as *cosine similarity* between client updates and a reference model, to identify and discard malicious contributions [7]. Our work explores two filtering techniques: an *angle-based filter*, which is a form of similarity check against the server model's gradient, and a *loss-based filter* inspired by prior work, which assesses a client's update based on the server's loss function.

FL with Server Data: The concept of leveraging a dataset at the server is a promising direction for improving FL, especially in non-IID environments. A server-side dataset can be used for various purposes, including regularization, knowledge distillation, and as a reference for detecting malicious behavior [8]. In some approaches, the server is treated as a special learning client. In contrast to work that assumes that the server

[1]V. S. Mai is with National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899, USA vansy.mai@nist.gov

[2]R. J. La is with the University of Maryland, College Park, MD 20742, USA hyongla@umd.edu

[3]K. Chakrabarti is with Tata Consultancy Services Research, Mumbai, Maharashtra 400607, India chakrabarti.k@tcs.com

[4]D. Maity is with the University of North Carolina at Charlotte, NC 28223, USA dmaity@charlotte.edu

The work of D. Maity was supported in part by the grant ARL DCIST CRA W911NF-17-2-0181 and the NSF CAREER Award 2443349.

data has the same distribution as the client data, we consider a more realistic scenario where the server's dataset can be small, synthetic, or have a different distribution. It builds on the idea that even a small, imperfect dataset at the server can guide the global model toward a better solution and aid in identifying harmful updates. This is particularly relevant in the context of Byzantine attacks, where malicious updates can be viewed as extreme cases of out-of-distribution data.

The novelty of our work lies in the synergistic combination of these three areas. By using the server's model not just for regularization but also as an active component to filter malicious updates, the proposed method can achieve an "honest majority" condition even when more than half of the participating clients are malicious. This integration of SL with robust aggregation and filtering presents a *complementary* approach to enhancing reliability of FL under Byzantine threats. We focus on presenting our method and experimental results in this paper, deferring theoretical analysis to future work.

The rest of the paper is as follows. The problem formulation and our proposed approach are given in Section II. The main algorithm is described in Section III, followed by experimental evaluations in Section IV.

Notation: For an integer $n > 0$, let $[n] := \{1, \dots, n\}$. For a finite set $\mathcal{D}$, $|\mathcal{D}|$ denotes its cardinality. For $x \in \mathbb{R}^n$, $\|x\|$ denotes its 2-norm. The inner product of x and y is denoted by $\langle x, y \rangle$.

II. Problem Formulation and Our Approach

A. Problem Formulation

Consider N clients with their datasets denoted by $\{\mathcal{D}_i\}_{i=1}^N$. Each client i uses $f_i(x) = \frac{1}{n_i} \sum_{s \in \mathcal{D}_i} \ell(x, s)$ with $n_i = |\mathcal{D}_i|$ as the local loss function over dataset $\mathcal{D}_i$ under a model x and sample loss ℓ. The goal of FL is to find

$$x^* \in \arg\min_{x \in \mathbb{R}^d} \; F(x) = \sum_{i \in [N]} p_i f_i(x), \tag{1}$$

where $(p_i)_{i \in [N]}$ is a probability vector, which is usually given by $p_i = \frac{n_i}{n}$ for all $i \in [N]$ with $n = \sum_{i \in [N]} n_i$. The main role of the server is to coordinate clients' learning, but it can also incorporate additional regularization as follows:

$$\min_{x \in \mathbb{R}^d} \quad \gamma f_0(x) + F(x) \tag{2}$$

where f_0 is the regularizer. We are interested in the case where there are malicious clients, denoted by $\mathcal{B}$, whose goal is to prevent honest clients from learning a good model by injecting bad updates in the FL process. Additionally, the server has access to some related representative data, denoted by $\mathcal{D}_0$, which could be synthetic or real data collected either from test clients or beta users, and uses it for learning with a loss function $f_0(x) = \frac{1}{n_0} \sum_{s \in \mathcal{D}_0} \ell(x, s)$. Thus, our goal becomes

$$\min_{x \in \mathbb{R}^d} \quad \gamma f_0(x) + F'(x), \tag{3}$$

where $F'(x) = \sum_{i \in [N] \setminus \mathcal{B}} p_i f_i(x)$.

B. Proposed Approach

We will use server data not just for filtering but also for learning to enhance the robustness of FL. The main idea is that when the distribution of server data $\mathcal{D}_0$ and that of honest clients $\mathcal{D}' = \cup_{i \in [N] \setminus \mathcal{B}} \mathcal{D}_i$ are close, server's loss function f_0 will be similar to the overall loss F' in (3). Thus, when far from convergence, the gradient ∇f_0 will more or less track the global gradient $\nabla F'$, even when individual clients' gradients ∇f_i including malicious ones do not follow $\nabla F'$ closely. This fact can be employed in two ways. First, updates from clients that significantly increase or do not improve server's loss f_0 can be regarded as unuseful or malicious and thus should be filtered out. Second, when the updated model obtained by aggregating client updates does not make much progress, ∇f_0 will help improve the updated model. In fact, as shown in [4], significant improvements can still be achieved even when the distributions of $\mathcal{D}_0$ and $\mathcal{D}'$ are not very similar as long as their difference is small in relation to the divergence in the distributions of clients' datasets; this will likely hold for malicious clients. By treating the server as an honest learner and implementing incremental server filtering and learning, we can effectively achieve an "honest majority", even when the number of malicious clients exceeds half. Our approach can be considered as a combination of [4] and [3].

III. Main Algorithm

Our approach, termed RoFSL, is presented in Algorithm 1. Lines 1–7 of RoFSL follow the FedAvg algorithm [9], where in each global round t, a selected client i, if not a malicious one, receives the current global model x_t from the server, performs K steps of the Stochastic Gradient Descent (SGD) algorithm using its local data $\mathcal{D}_i$ (LocalSGD), and returns to the server its latest update. On the other hand, if client i is malicious, its reported update can be anything. The server then combines all received updates using a robust aggregation and uses the resulting model to learn locally by performing K_0 steps of LocalSGD. We now describe these steps of our algorithm RoFSL in detail below.

1) Robust Aggregation: As we mentioned above, when far from convergence, the gradient ∇f_0 will more or less track the global gradient $\nabla F'$. This fact can be employed to filter out updates from malicious clients using the following ideas.

*a) Angle-based filtering (*AF*):* Client updates $\Delta x_t^{(i)}$ that are too far from the direction of $-\nabla f_0(x_t)$ deem unuseful or likely malicious. This can be done by using a threshold α on their cosine similarity, defined as $\text{cos\_sim}(x, y) = \frac{\langle x, y \rangle}{\|x\| \|y\|}$ for any $x, y \in \mathbb{R}^d$. For $\mathcal{S} \subset [N]$, define

$$\text{AF}_\alpha(\mathcal{S}) := \{i \in \mathcal{S} : \quad \text{cos\_sim}(\Delta x^{(i)}, -\nabla f_0(x)) \geq \alpha\}.$$

Choosing an appropriate threshold is difficult: a large α may exclude benign updates, while a small α risks including malicious ones. Since this filter only considers direction, malicious clients could use large updates near the threshold to disrupt convergence. Thus, we adopt a loose threshold such as $\alpha = 0$ combined with robust aggregation to ensure effectiveness.

*b) Loss-based filtering (*LF): Define the following score

$$\text{sc}_\rho^{(i)}(x) = -\langle \Delta x^{(i)}, \nabla f_0(x) \rangle - \rho \|\Delta x^{(i)}\|^2$$

as a second-order approximation to $f_0(x) - f_0(x + \Delta x^{(i)})$, i.e., the improvement of the server loss when using client i's model. This score is related to the cosine similarity in AF through the inner product term. But, it also takes into account the magnitude of the update penalizing large deviations via $-\rho\|\Delta x^{(i)}\|^2$. This score is also related to the stochastic descent score in [3], where the authors consider $\Delta x^{(i)}$ to be (negative of) the stochastic gradient of client i, normalized to have the same norm as $\nabla f_0(x)$, and use a fixed threshold ϵ to filter clients' models based on their scores, e.g., $\text{sc}_\rho^{(i)}(x) \geq -\epsilon$. Again, choosing a suitable threshold ϵ (even just the range of it) is nontrivial as the fraction of malicious clients and the non-IIDness of client data are unknown. In our case, we allow $\Delta x^{(i)}$ to be the client update (after a number of local training steps without normalizing) and consider their scores as defined above for filtering. In general, one could consider clustering the scores, but for simplicity, we filter out a fixed fraction of updates with the lowest scores, denoted by $\theta \in (0, 1)$, in every round. Specifically, for any set $\mathcal{S}$, we first sort $\text{sc}_\rho^{(i_1)}(x) \leq \text{sc}_\rho^{(i_2)}(x) \leq \ldots \leq \text{sc}_\rho^{(i_{|\mathcal{S}|})}(x)$ and then find

$$\texttt{LF}_{\rho,\theta}(\mathcal{S}) = \{i_{\lceil \theta \times |\mathcal{S}| \rceil}, \ldots, i_{|\mathcal{S}|}\} \subset \mathcal{S}$$

as the set of accepted clients. For example, choosing $\theta = 0.5$ removes half of the sampled clients in each round if we believe that the majority of clients is honest. Of course, it is unlikely that the fraction of malicious clients is known in advance and this assumption may not hold. Furthermore, even if the fraction of malicious clients is known and less than 0.5, since $\mathcal{S}$ is randomly selected in each round, the majority of sampled clients in a round can be malicious. Thus, a robust aggregation step is still needed after this filtering step.

c) Model aggregation: We chose geometric median as it has been shown to be relatively robust, despite the server's increased computational cost. In fact, although

$$\texttt{GeoMed}(x_i, i \in \mathcal{S}) \in \arg\min_x \sum_{i \in \mathcal{S}} \|x - x_i\|$$

has no closed-form solution, an approximate solution can be computed efficiently using numerical methods such as a Weiszfeld-type algorithm [2] or an interior-point method [10].

d) Norm clipping: Even with filtering and robust aggregation, malicious updates can still negatively influence the combined model if they constitute a majority. Thus, a clipping step is applied after aggregation to restrict potential damage: $\texttt{Clip}_\tau(x) = \min\left(1, \tau/\|x\|\right) \times x$, where $\tau > 0$ is a clipping parameter set by the server. This is different from gradient clipping in local training steps of the clients. It also presents a similar trade-off for robustness as choosing conservative τ would also limit the contribution of honest clients. Our aim is to use SL to offset such a limit. As a result, line 7 of our algorithm (RoFSL in Algorithm 1) performs the following:

$$\texttt{RobustAggr} \equiv \texttt{Clip}_\tau \circ \texttt{GeoMed} \circ \texttt{Filter},$$

where Filter can be $\texttt{AF}_\alpha$, $\texttt{LF}_{\rho,\theta}$, or 0F (neither).

Algorithm 1: RoFSL: Robust FL via Server Learning

Server: initial x_0, learning rates η_g, η_0, no. steps K_0, weight γ
Clients: learning rate η_l, no. steps K

1 **for** $t = 0, \ldots, T-1$ **do**
2 sample a subset $\mathcal{S}$ of clients
3 broadcast x_t to clients in $\mathcal{S}$
4 **forall** clients $i \in \mathcal{S}$ **do**
5 $x_t^{(i)} \leftarrow \begin{cases} \texttt{LocalSGD}(f_i, x_t, \eta_l, K), & i \notin \mathcal{B} \\ \texttt{Attack} & i \in \mathcal{B} \end{cases}$
6 upload $x_t^{(i)} \rightarrow$ Server
7 $\bar{x}_t \leftarrow \texttt{RobustAggr}\big(x_t^{(i)}, i \in \mathcal{S}\big)$
8 $x_{t+1} \leftarrow \texttt{LocalSGD}(\gamma f_0, \bar{x}_t, \eta_0, K_0)$

$\texttt{LocalSGD}(f, x, \eta, K)$:
9 $y_0 \leftarrow x$
10 **for** $k = 0, \ldots, K-1$ **do**
11 $g(y_k) \leftarrow$ unbiased estimate of $\nabla f(y_k)$
12 $y_{k+1} \leftarrow y_k - \eta g(y_k)$
13 **return**: y_K

2) Server Learning: In the case without attacks, [4] shows that SL can significantly improve convergence. In the presence of Byzantine clients, we believe the role of SL is even more significant as it can be viewed as an honest learner providing (i) a direction for update filtering as shown above and (ii) enhancement to model convergence. More importantly, by employing incremental SL, we can practically achieve the "honest majority" condition without requiring more than half of the clients to be honest. Since the server data can be small and different from the aggregated data distribution of honest clients, we do not want to over-utilize SL to avoid overfitting server data; this can be achieved by tuning the weight γ in (3). In addition, we impose the same norm-clipping step at the server to further limit such overfitting while providing enough room for the server's update to act as correction to the aggregated model from clients. Finally, SL in [4] adopts a "pseudo-gradient" step with a step size of $\eta_g > 1$ using the averaged update from clients prior to the SL steps, namely,

$$\bar{x}_t \leftarrow x_t + \eta_g \textstyle\sum_{i \in \mathcal{S}} (x_t^{(i)} - x_t)/|\mathcal{S}|.$$

In Algorithm 1, we ignore this "pseudo-gradient" step (by setting $\eta_g = 1$ and using RobustAggr described above) because we believe it can potentially amplify the effect of malicious updates and reduce the robustness of the algorithm. We will illustrate this point in our experiments below.

IV. Experimental Results

We illustrate the benefits of RoFSL through experiments using EMNIST [11] and CIFAR-10 [12]. We evaluate robustness under non-IID settings while varying the fraction of Byzantine clients performing an equal mix of sign-flipping and label-flipping attacks. Sign-flipping directly corrupts model updates, while label-flipping represents a data-poisoning attack whose biased updates are harder to distinguish from honest ones in the presence of non-IID data. By combining these complementary attack types and varying the Byzantine fraction, we

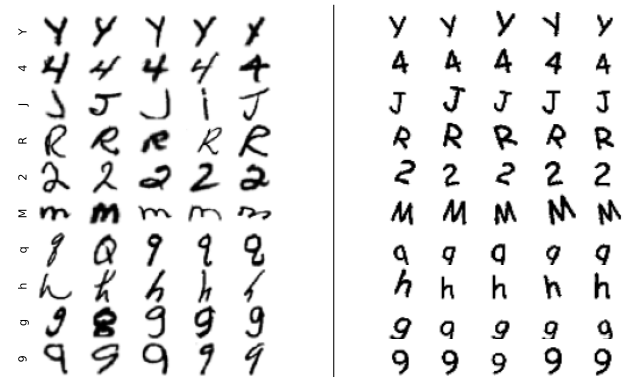

Fig. 1. Left: EMNIST training examples. Right: Server's synthetic examples.

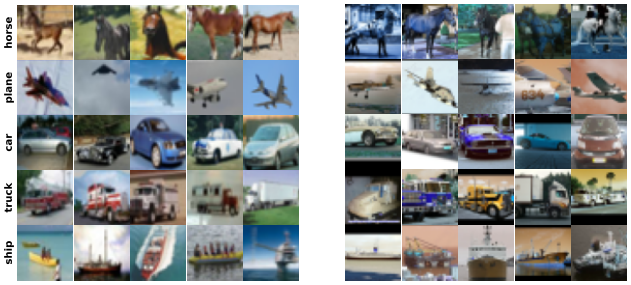

Fig. 2. Left: CIFAR-10 training examples. Right: Server's STL-10 examples.

obtain a comprehensive and realistic assessment of robustness across both data heterogeneity and adversarial intensity. More experimental results can be found in [13].

A. Setup

1) Data: For CIFAR-10, we use 50k samples for training and 10k for testing. For EMNIST, we use a balanced dataset with 45 classes, 108k samples for training and 18k for testing. Training data is split among clients in a non-IID fashion.

For server, we consider synthetic data or data from a different source: for EMNIST, we provide the server $n_0 = 900$ *synthetic* examples by generating for each label class 20 images of the corresponding letter or number using a cursive font with 5 rotation angles $\{-20, -10, 0, 10, 20\}$ and 4 sizes;[1] see Fig. 1 for a comparison of this synthetic data and EMNIST. We note that many classes in the balanced EMNIST dataset include both upper-case and lower-case letters whereas the synthetic data at the server only contains a single case for each class (e.g., `J` and `M` shown in Fig. 1). For CIFAR-10, we collect $n_0 = 900$ images from the dataset STL-10 with 9 similar label classes as in CIFAR-10, each with 100 examples;[2] see Fig. 2 for an illustration of this data, and note that the class `frog` is absent in STL-10.

In both cases, the server data exhibits shortcomings of synthetic data that one often expects in practice, namely limited quantity, different distributions, and possibly missing data or partially correct labeled data. We do not attempt to improve the quality and quantity of server data; our goal is to examine the benefits of SL when it is performed on data with a significantly different distribution than that of clients' data.

2) Models: We use simple neural networks with 2 convolutional layers (with a kernel size of 3 and `padding=1`) and 2-3 dense layers. All layers use `ReLU` as activation functions except for the last dense layer that uses `softmax`. For *EMNIST* dataset, we use Conv2d(1,32)–Conv2d(32,64)–MaxPool2d(2)–Dropout(0.25)–Dense(128)–Dropout(0.5)–Dense(45), and for *CIFAR-10*, Conv2d(3,32)–MaxPool2d(2)–Conv2d(32,64)–MaxPool2d(2)–Dropout(0.25)–Dense(128)–Dropout(0.5)–Dense(10). Note that these models are by no means the state-of-the-art.

3) Attacks: In each run, we assign a fraction β of clients, selected uniformly at random (u.a.r.) without replacement to be attackers and consider an equal mix of 2 attack types, namely sign-flipping and label-flipping, denoted by $\mathcal{B}_s$ and $\mathcal{B}_l$, respectively. This ensures that the robust aggregation method is not tuned or accidentally specialized to a single type.

a) Sign-flipping: Attackers train their models as usual to obtain $\Delta x_t^{(i)}$ but send server $\Delta \check{x}_t^{(i)} = -\nu_i \Delta x_t^{(i)}$, or equivalently, $\check{x}_t^{(i)} = x_t - \nu_i \Delta x_t^{(i)}$, where $\nu_i > 0$ controls the strength of the attack selected u.a.r. in $[0.1, 10.1]$ for all $i \in \mathcal{B}_s$.

b) Label-flipping: Attackers shift their (encoded) labels by 1 (i.e., class $c \to c+1$) and scale their learning rate by a factor ν_i selected u.a.r. in $[0.1, 2.1]$ for $i \in \mathcal{B}_l$. In each round, attackers train their models and return their updates.

4) Implementation and Evaluation: We partition training data roughly evenly among N clients but control their non-IIDness by selecting the Dirichlet distribution parameter in $\{0.1, 0.3\}$; here $N = 450$ for EMNIST and $N = 1000$ for CIFAR-10. In each round, only $S = 20$ clients are sampled u.a.r. without replacement. Honest clients sampled in each round train their local models for $E_c = 2$ epochs on their local data with batch size B, where $B = 50$ for EMNIST and $B = 25$ for CIFAR-10. The server also updates its model for $E_0 = 2$ epochs using batch size $B_0 = 180$ (i.e., 10 local steps per round). Additionally, for local training, we use cross-entropy loss with a weight decay of 10^{-4} for EMNIST and 10^{-3} for CIFAR-10. We employ the usual SGD method with a fixed learning rate $\eta = 0.1$ in all cases. The number of global rounds is set to be 500 for EMNIST and 1500 for CIFAR-10.

For our robust aggregation step, we use the following parameters as default: $\alpha = 0$ for $\texttt{AF}_\alpha$, $(\rho, \theta) = (0.1, 0.5)$ for $\texttt{LF}_{\rho,\theta}$, and $\tau = 1$ for $\texttt{Clip}_\tau$. For `GeoMed`, we use the smooth version of Weiszfeld's algorithm in [2] with a maximum of 4 steps and a relative objective error of 10^{-6} as stopping criteria.

Finally, we use accuracy as the main metric as test data is balanced. Reported numbers are the averages of 3 runs and a rolling window of size 20 for EMNIST and 50 for CIFAR-10.

B. Results

We first verify that training on the server data alone does not provide significant benefits by using a grid search with bach sizes $\{100, 200\}$, learning rates $\{0.002, 0.005, 0.01, 0.1\}$, epochs $\{200, 400\}$, and optimizers $\{\texttt{SGD}, \texttt{Adam}\}$. The highest test accuracy is about 22% for both datasets. In experiments

[1] We first plot each character or number in a 2 inch $\times$ 2 inch figure using font sizes $\{100, 110, 120, 130\}$ in points with each point equal to $1/72$ inch, and then resize it to a 28 pixel $\times$ 28 pixel figure.

[2] Data available at: https://cs.stanford.edu/~acoates/stl10/

TABLE I

ACCURACIES AFTER 500 ROUNDS FOR EMNIST AND 1500 ROUNDS FOR CIFAR10 WITH DIFFERENT PARAMETERS OF SERVER LEARNING γ, BYZANTINE FRACTION β, AND FILTERING: NO FILTER (0F), ANGLE-BASED FILTER (AF), AND LOSS-BASED FILTER (LF).

EMNIST		$\beta = 0$			$\beta = 0.3$			$\beta = 0.6$		
SL	γ	0F	AF	LF	0F	AF	LF	0F	AF	LF
EMNIST Dir(0.1)	0.0	$84.96_{\pm 0.18}$	$83.46_{\pm 0.57}$	$84.11_{\pm 0.26}$	$81.47_{\pm 0.58}$	$69.12_{\pm 4.33}$	$67.07_{\pm 4.64}$	$4.78_{\pm 2.02}$	$14.31_{\pm 3.53}$	$2.22_{\pm 0.00}$
	.05	$84.89_{\pm 0.15}$	$84.28_{\pm 0.17}$	$83.70_{\pm 0.35}$	$81.68_{\pm 0.50}$	$82.25_{\pm 0.24}$	$83.45_{\pm 0.18}$	$7.53_{\pm 2.69}$	$29.03_{\pm 3.90}$	$74.41_{\pm 2.15}$
	0.1	$84.91_{\pm 0.11}$	$84.10_{\pm 0.21}$	$83.45_{\pm 0.29}$	$81.74_{\pm 0.42}$	$82.57_{\pm 0.26}$	$83.34_{\pm 0.14}$	$7.80_{\pm 2.77}$	$37.67_{\pm 4.50}$	$77.32_{\pm 1.51}$
	0.2	$84.82_{\pm 0.17}$	$84.00_{\pm 0.18}$	$83.38_{\pm 0.28}$	$81.61_{\pm 0.44}$	$82.08_{\pm 0.24}$	$83.29_{\pm 0.27}$	$8.01_{\pm 2.82}$	$48.94_{\pm 4.20}$	$76.93_{\pm 2.02}$
	0.5	$84.83_{\pm 0.12}$	$83.89_{\pm 0.26}$	$83.17_{\pm 0.19}$	$81.47_{\pm 0.33}$	$81.34_{\pm 0.37}$	$83.08_{\pm 0.25}$	$8.91_{\pm 2.82}$	$42.89_{\pm 2.73}$	$76.44_{\pm 2.36}$
	1.0	$84.78_{\pm 0.11}$	$83.68_{\pm 0.25}$	$83.17_{\pm 0.25}$	$81.53_{\pm 0.35}$	$80.60_{\pm 0.41}$	$83.02_{\pm 0.24}$	$9.11_{\pm 3.02}$	$41.31_{\pm 2.10}$	$75.91_{\pm 2.66}$
	2.0	$84.78_{\pm 0.11}$	$83.62_{\pm 0.25}$	$83.18_{\pm 0.28}$	$81.44_{\pm 0.37}$	$80.79_{\pm 0.36}$	$83.05_{\pm 0.20}$	$8.94_{\pm 3.03}$	$39.54_{\pm 2.65}$	$75.96_{\pm 2.47}$
EMNIST Dir(0.3)	0.0	$85.75_{\pm 0.14}$	$84.07_{\pm 0.44}$	$84.93_{\pm 0.26}$	$82.95_{\pm 0.48}$	$71.07_{\pm 8.24}$	$71.40_{\pm 6.77}$	$4.14_{\pm 3.76}$	$19.03_{\pm 8.84}$	$2.22_{\pm 0.00}$
	.05	$85.74_{\pm 0.12}$	$85.13_{\pm 0.25}$	$84.92_{\pm 0.15}$	$82.85_{\pm 0.37}$	$83.73_{\pm 0.28}$	$84.67_{\pm 0.19}$	$5.88_{\pm 4.12}$	$30.91_{\pm 7.58}$	$20.73_{\pm 6.22}$
	0.1	$85.65_{\pm 0.10}$	$85.03_{\pm 0.26}$	$84.74_{\pm 0.20}$	$82.87_{\pm 0.31}$	$83.67_{\pm 0.25}$	$84.56_{\pm 0.18}$	$6.01_{\pm 3.96}$	$31.69_{\pm 9.02}$	$74.66_{\pm 4.55}$
	0.2	$85.63_{\pm 0.10}$	$84.29_{\pm 0.32}$	$84.61_{\pm 0.15}$	$82.94_{\pm 0.25}$	$81.70_{\pm 0.41}$	$84.53_{\pm 0.16}$	$6.37_{\pm 3.89}$	$52.90_{\pm 5.92}$	$79.86_{\pm 1.85}$
	0.5	$85.52_{\pm 0.10}$	$68.39_{\pm 0.42}$	$84.55_{\pm 0.13}$	$82.82_{\pm 0.22}$	$35.05_{\pm 0.24}$	$84.37_{\pm 0.14}$	$6.52_{\pm 3.89}$	$20.53_{\pm 0.00}$	$79.22_{\pm 2.27}$
	1.0	$85.49_{\pm 0.09}$	$61.11_{\pm 0.15}$	$84.48_{\pm 0.09}$	$82.80_{\pm 0.24}$	$35.56_{\pm 0.00}$	$84.39_{\pm 0.14}$	$6.75_{\pm 4.07}$	$20.21_{\pm 0.00}$	$79.59_{\pm 2.47}$
	2.0	$85.48_{\pm 0.08}$	$82.92_{\pm 0.23}$	$84.52_{\pm 0.16}$	$82.73_{\pm 0.23}$	$52.70_{\pm 0.34}$	$84.33_{\pm 0.14}$	$6.60_{\pm 4.00}$	$23.87_{\pm 0.00}$	$78.79_{\pm 2.62}$
CIFAR-10 Dir(0.1)	0.0	$71.40_{\pm 0.60}$	$61.87_{\pm 2.24}$	$67.38_{\pm 1.13}$	$61.36_{\pm 1.51}$	$43.63_{\pm 3.61}$	$52.03_{\pm 2.85}$	$11.78_{\pm 1.50}$	$11.07_{\pm 1.51}$	$11.81_{\pm 0.95}$
	.05	$71.60_{\pm 0.44}$	$69.32_{\pm 0.81}$	$68.36_{\pm 1.06}$	$62.07_{\pm 0.91}$	$62.38_{\pm 1.76}$	$65.68_{\pm 1.28}$	$12.04_{\pm 1.32}$	$12.18_{\pm 1.69}$	$35.95_{\pm 2.69}$
	0.1	$71.06_{\pm 0.34}$	$68.70_{\pm 0.87}$	$67.77_{\pm 0.65}$	$61.53_{\pm 1.15}$	$61.81_{\pm 1.40}$	$64.72_{\pm 1.08}$	$12.61_{\pm 1.16}$	$16.09_{\pm 0.93}$	$44.60_{\pm 4.82}$
	0.2	$70.83_{\pm 0.38}$	$68.04_{\pm 0.80}$	$66.88_{\pm 0.72}$	$60.93_{\pm 0.97}$	$59.31_{\pm 1.90}$	$62.13_{\pm 0.84}$	$13.01_{\pm 1.48}$	$14.11_{\pm 1.51}$	$38.54_{\pm 4.64}$
	0.5	$69.97_{\pm 0.45}$	$67.11_{\pm 1.00}$	$65.97_{\pm 0.93}$	$60.08_{\pm 1.15}$	$58.89_{\pm 0.84}$	$62.10_{\pm 1.02}$	$12.50_{\pm 1.40}$	$14.67_{\pm 0.78}$	$35.44_{\pm 3.30}$
	1.0	$69.18_{\pm 0.43}$	$66.42_{\pm 0.75}$	$65.83_{\pm 0.51}$	$60.01_{\pm 1.03}$	$57.33_{\pm 1.83}$	$61.69_{\pm 0.98}$	$12.45_{\pm 1.18}$	$16.16_{\pm 1.81}$	$35.11_{\pm 3.31}$
	2.0	$67.68_{\pm 0.60}$	$64.82_{\pm 0.92}$	$64.06_{\pm 0.93}$	$58.14_{\pm 1.02}$	$56.06_{\pm 1.13}$	$59.61_{\pm 0.50}$	$12.65_{\pm 1.30}$	$17.70_{\pm 1.79}$	$32.96_{\pm 3.12}$
CIFAR-10 Dir(0.3)	0.0	$73.82_{\pm 0.37}$	$69.28_{\pm 0.69}$	$71.72_{\pm 0.76}$	$66.12_{\pm 1.17}$	$43.18_{\pm 7.94}$	$56.56_{\pm 7.78}$	$8.98_{\pm 1.58}$	$13.11_{\pm 2.26}$	$9.87_{\pm 0.97}$
	.05	$74.10_{\pm 0.11}$	$72.85_{\pm 0.34}$	$71.63_{\pm 0.54}$	$66.32_{\pm 1.11}$	$67.34_{\pm 0.91}$	$70.27_{\pm 0.46}$	$9.27_{\pm 1.39}$	$10.94_{\pm 3.04}$	$53.58_{\pm 5.77}$
	0.1	$73.53_{\pm 0.10}$	$72.03_{\pm 0.20}$	$71.53_{\pm 0.41}$	$65.49_{\pm 0.76}$	$66.75_{\pm 0.52}$	$69.88_{\pm 0.52}$	$9.89_{\pm 1.34}$	$12.37_{\pm 2.28}$	$52.89_{\pm 6.79}$
	0.2	$72.89_{\pm 0.09}$	$71.35_{\pm 0.35}$	$70.93_{\pm 0.25}$	$64.68_{\pm 0.95}$	$64.49_{\pm 0.96}$	$67.97_{\pm 0.76}$	$10.14_{\pm 1.14}$	$11.60_{\pm 1.81}$	$42.74_{\pm 8.24}$
	0.5	$72.60_{\pm 0.14}$	$70.70_{\pm 0.50}$	$70.38_{\pm 0.35}$	$64.20_{\pm 0.82}$	$62.66_{\pm 1.67}$	$66.97_{\pm 0.57}$	$10.06_{\pm 1.20}$	$12.65_{\pm 1.36}$	$43.26_{\pm 5.36}$
	1.0	$72.11_{\pm 0.07}$	$70.13_{\pm 0.66}$	$69.52_{\pm 0.49}$	$64.18_{\pm 0.72}$	$62.42_{\pm 1.44}$	$66.49_{\pm 0.40}$	$9.77_{\pm 1.44}$	$12.80_{\pm 1.84}$	$39.27_{\pm 6.68}$
	2.0	$70.71_{\pm 0.11}$	$69.02_{\pm 0.45}$	$67.97_{\pm 0.58}$	$63.21_{\pm 0.80}$	$61.84_{\pm 1.11}$	$64.61_{\pm 0.60}$	$9.62_{\pm 1.36}$	$14.43_{\pm 3.44}$	$39.82_{\pm 4.98}$

below, we ignore this pretrained model to avoid potential overfitting on server data, start with a random model and use SGD. We vary the SL parameter $\gamma \in [0, 2]$ and the fraction of malicious clients $\beta \in [0, 0.6]$. The results are given in Table I.

1) SL without Server Filtering (SF): Without attacks, i.e., $\beta = 0$, adding SL does not seem to improve accuracy. This is because of the following two reasons. First, the server data is rather different from the clients' training data. Second, unlike FSL that adopts a pseudo-gradient step with a step size $\eta_g > 1$ using the aggregated update from clients prior to SL [4], here we ignore this step (with $\eta_g = 1$) throughout our experiments because we believe this step can potentially amplify the effect of malicious updates and reduce the robustness of the algorithm. Additional results in [13] confirm this intuition.

2) SF without SL: This corresponds to the rows with $\gamma = 0$ in Table I. Here, except for EMNIST with $\beta = 0$, adding filters (either AF or LF) can reduce the accuracy considerably for both datasets with $\beta = 0$ or $\beta = 0.3$. To support this observation, we show in Fig. 3 the performance comparison for a finer range of $\beta \in [0, 0.7]$. This may seem rather counter-intuitive as one often expects the filtering step, when tuned properly, to reduce the number of corrupted updates for the aggregation step, thereby improving the aggregated model. We

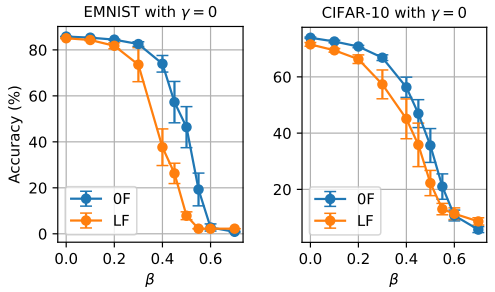

Fig. 3. Accuracy when using GeoMed but *without* SL.

attribute this effect to the GeoMed-aggregation step, which is known to have a break point of 0.5, i.e., it can provide a robust estimator when up to half of the data points are corrupted [14]. As a result, when β is small, adding filtering steps (without any finetuning) will likely remove updates from some honest clients (besides some malicious updates, especially those with significant attack strengths), leading to degradation of the geometric median of the remaining updates.

When $\beta = 0.6$, although AF appears to achieve slightly higher accuracy compared to 0F and LF, the performance of all schemes without SL degrades significantly. This is because the overwhelming presence of malicious clients sampled in

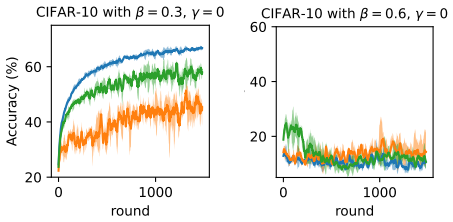

Fig. 4. Accuracy on CIFAR-10 Dir(0.3) using GeoMed but *without* SL.

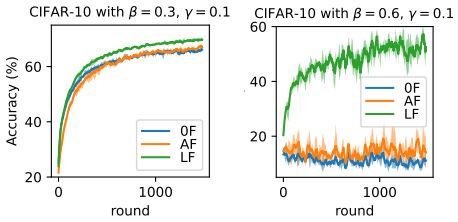

Fig. 5. Accuracy on CIFAR-10 Dir(0.3) using GeoMed and $\gamma = 0.1$ for SL.

most rounds makes it harder to filter out malicious updates and, as a result, many undesirable updates are used in the model aggregation steps, leading to poor performance. Fig. 4 shows the accuracy of the algorithm for CIFAR-10 $\texttt{Dir}(0.3)$.

3) Combining SL and SF: As we saw earlier, either SF or SL alone is not enough to counter malicious updates even when using GeoMed aggregation. However, when combined together, significant improvement can be achieved for large β as shown in Table I. Not surprisingly, we must use sufficiently large γ for SL to take effect: γ between $[0.05, 1.0]$ seems to be effective, suggesting that tuning this value is not difficult in practice. Fig. 5 shows the test accuracy when training on CIFAR-10 Dirichlet 0.3 using SL with $\gamma = 0.1$. When $\beta = 0.3$, LF provides significant improvement in terms of convergence rate and final accuracy. In fact, LF works even when $\beta = 0.6$ (while 0F fails to achieve any learning), suggesting improved robustness of our algorithm.

Note that in the case of $\texttt{AF}_0$, certain values of γ and Dirichlet parameter can cause a large accuracy drop (e.g., $\gamma = 0.5, 1$ in EMNIST-Dirichlet 0.3) as learning stops when the updated model falls into a local *trap* of server's loss function, where updates from sampled clients are filtered out after a while (explaining 0's in some reported deviations). This does not happen to $\texttt{LF}_{0.1,0.5}$ as we fix the fraction of filtered updates. Because of such phenomenon and the superior performance of LF over AF, we will focus on LF below.

Next, we vary the fraction of malicious clients for a fixed value of SL to see how the performance degrades. Fig. 6 shows the final accuracy of RoFSL for $\beta \in [0, 0.7]$, demonstrating a significant improvement in robustness of LF for both datasets.

4) Effect of Filter Parameters: We focus on LF as it shows notable improvements compared to AF. Fig. 7 shows the sensitivity of the final accuracy against the parameter ρ in $\texttt{LF}_{\rho,\theta}$. For both datasets, RoFSL achieves a consistent performance for $\rho \in [0.1, 1]$, suggesting that tuning this parameter should not be a significant challenge in practice.

5) Non-IIDness: Table I shows that in most cases, as expected, the final accuracy reduces as the client's data become more non-IID. The effect of server learning and filtering is fairly consistent across different datasets and Dirichlet settings.

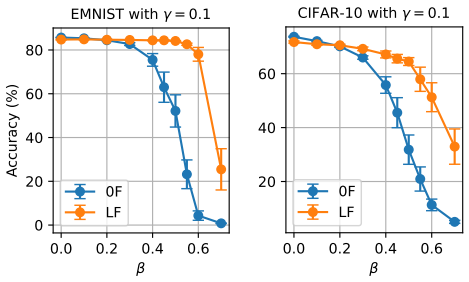

Fig. 6. Accuracy vs. malicious fraction β with GeoMed aggregation.

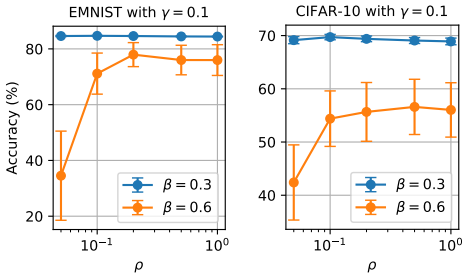

Fig. 7. Sensitivity against the parameter ρ of the loss-based filter LF.

References

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

[2] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 1142–1154, 2022.

[3] C. Xie, S. Koyejo, and I. Gupta, "Zeno++: Robust fully asynchronous SGD," in *ICML*, pp. 10495–10503, PMLR, 2020.

[4] V. S. Mai, R. J. La, and T. Zhang, "A study of enhancing federated learning on non-iid data with server learning," *IEEE Trans. AI*, vol. 5, no. 11, pp. 5589–5604, 2024.

[5] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *ICML*, pp. 6893–6901, PMLR, 2019.

[6] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "FLdetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proc. 28th ACM SIGKDD*, pp. 2545–2555, 2022.

[7] S. Awan, B. Luo, and F. Li, "Contra: Defending against poisoning attacks in federated learning," in *European symposium on research in computer security*, pp. 455–475, Springer, 2021.

[8] L. Li, J. Gou, B. Yu, L. Du, Z. Yi, and D. Tao, "Federated distillation: A survey," *arXiv preprint arXiv:2404.08564*, 2024.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artif. Intell. Stat. (AISTATS)*, pp. 1273–1282, PMLR, 2017.

[10] M. B. Cohen, Y. T. Lee, G. Miller, J. Pachocki, and A. Sidford, "Geometric median in nearly linear time," STOC '16, p. 9–21, 2016.

[11] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *Int. Jt. Conf. Neural Netw.*, pp. 2921–2926, 2017.

[12] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009.

[13] V. S. Mai, R. J. La, K. Chakrabarti, and D. Maity, "Enhancing Robustness of Federated Learning via Server Learning," *arXiv:2604.03226*, 2026.

[14] H. P. Lopuhaä and P. J. Rousseeuw, "Breakdown points of affine equivariant estimators of multivariate location and covariance matrices," *The Annals of Statistics*, pp. 229–248, 1991.