Abhijit Debnath
Uddalok Sen
Uddalak Seal
Abhoy Sarkar

ENHANCING PRIVACY AND ACCURACY IN FEDERATED LEARNING FOR REGRESSION WITH SERVER-SIDE FILTERING TO ADDRESS OUTLIERS

Abstract *Federated learning offers a solution to privacy-related dilemmas of data centralization by maintaining a decentralized architecture, thereby enabling local devices to preserve their data while concurrently exchanging model parameters. Despite its promise, federated learning encounters substantial obstacles concerning data quality, which may arise from inherent biases, the presence of outliers, and the utilization of compromized devices. To mitigate these challenges, we advocate for the implementation of a server-side filtering methodology within federated learning, specifically tailored for regression-related problems. Based on this architecture, local devices train the model on their own data sets and then send the learned parameters to a central server. The server is then tasked with the filtration of erroneous contributions, thereby enhancing the overall accuracy of the model. This methodology is substantiated through the application of the mean squared error (MSE) metric, a widely recognized standard within regression analysis, thereby augmenting both the efficiency and dependability of the learning process while safeguarding user privacy, an essential component of federated learning.*

Keywords federated learning, linear regression, server-side filter, data privacy, noise filtering

Citation Computer Science 27(1) 2026: 25–42

1. Introduction

The progression of artificial intelligence has been significantly propelled by the accessibility of extensive datasets derived from various sources, including sensors, mobile devices, and IoT devices. Nevertheless, the training of precise models necessitates substantial, frequently proprietary data sets that have traditionally been collected and stored in a centralized manner, which consequently raises significant privacy issues [12].

Federated learning presents a viable resolution by retaining data on localized devices while exclusively transmitting learning parameters to a central server, thereby safeguarding privacy [8] and [13]. Within this framework, local devices (clients) engage in model training, utilizing their respective data sets and subsequently relay parameters, namely weights and biases, to a server, which synthesizes these inputs to refine a global model until the model achieves convergence, satisfying predetermined criteria for accuracy, and communication rounds.

In addition to its inherent benefits, federated learning encounters limitations attributable to its decentralized architecture [7]. A client may possess biased, erroneous, or outlier data, which we classify as representing a malicious client. Data can indeed be biased, erroneous, or anomalous. During the training phase, these malicious clients transmit model parameters; the inclusion of such during aggregation adversely affects both model accuracy and convergence velocity, resulting in compromised outcomes [6]. Non-IID data distributions make handling these hostile clients even more difficult [11].

Our study offers a novel server-side filtering technique. Using MSE as a metric to evaluate model performance, our method successfully eliminates malicious and outlier parameters before aggregation when implemented in a federated linear regression framework. This method preserves the integrity of data privacy while also improving model accuracy. The following are the contributions made by this work.

For federated learning, we suggest a strong server-side filtering mechanism that offers significant benefits over conventional median-based methods. Our new filtering method increases model accuracy, which is essential for making accurate predictions in federated learning, especially when communication cycles are limited. By applying our approach before parameter aggregation, we guarantee that model parameters from malicious clients are excluded, improving the global model's correctness.

We propose a novel server-side filtering approach to address important issues in federated learning environments. The suggested approach uses a federated-linear-regression framework and a strict filtering procedure to remove malicious and abnormal parameters before the aggregation stage. Since this approach preserves data secrecy while also increasing the global model's accuracy, it marks a substantial breakthrough in the field of federated learning.

The main contributions of this study include:

- **Construction of a robust server-side filtering system:** Our approach goes beyond traditional median-based methods, guaranteeing improved resilience to

hostile and noisy client updates, also improved accuracy of models in federated learning: Significant improvements in predicted accuracy are demonstrated by the suggested filtering technique, which is particularly important in scenarios with few communication rounds.

- **The incorporation of pre-aggregation filtering:** By integrating this technique within the parameter aggregation procedure, the methodology effectively diminishes the impact of malicious clients, thereby optimizing the performance of the global model.

The structure of the manuscript is as follows: While Section 3 provides further details on the federated learning framework and our suggested method, Section 2 examines relevant work. Section 4 presents the results of the simulation and the performance analysis; Section 5 is provided for further discussion of FL's clients. Lastly, Section 6 wraps up the work by offering suggestions for possible future research avenues.

2. Related literature

This section includes the literature of previous works related to this study and the comparison of works. A table of comparison is also provided for better understanding of the nature of the works. The study is divided into different subsections.

2.1. Distributed and federated learning and challenges

Distributed learning constitutes a machine learning paradigm wherein the training procedure is disseminated across multiple computational nodes. This framework is exceptionally efficacious for managing extensive datasets and intricate models that would otherwise be computationally unmanageable for a singular machine. The fundamental concept involves partitioning data and computational tasks among nodes, facilitating parallel processing and diminishing computation duration.

A specific type of distributed learning is called federated learning (FL), in which clients work together to create a global model using their locally stored data. Clients train the models locally instead, then send just the parameters they have learned to a central server. After refining the global model by combining these factors the server redistributes it to the clients. Iteratively, this exchange continues until the global model converges. FL prioritizes privacy as a foundational design principle. This distinction engenders unique challenges and advantages, particularly in contexts involving sensitive or decentralized data.

Communication overhead FL necessitates frequent exchanges of model parameters between clients and the server, which can incur considerable costs in terms of bandwidth and latency. Strategies aimed at minimizing communication overhead include extending client-side training duration [8], although this introduces computational demands at the client level.

Data heterogeneity clients operating within a federated learning framework frequently function in heterogeneous environments characterized by diverse data quality and volume. Both model accuracy and convergence are hampered by this non-IID (non-identically and independently distributed) data distribution.

2.2. Privacy preservation techniques

Secure multiparty computation (SMPC): Through encryption, this technique protects data while guaranteeing that clients retain control over their local datasets. [3] presented a framework for SMPC-based privacy-preserving federated learning; nevertheless, this method has a significant communication overhead.

In order to prevent potential attackers from inferring sensitive information, differential privacy (DP) uses stochastic noise to obfuscate data. While effective in enhancing privacy, this method may negatively impact model performance due to noise-induced inaccuracies [5].

However, there is a significant computational cost associated with the encryption procedures [15]. Finding a balance between client-side calculations and communication rounds is essential for effective strategies to reduce communication overhead [9]. One way to reduce communication frequency is to increase local training epochs before parameter exchanges, although this requires careful optimization to maintain model accuracy. There are many difficulties, since client data in federated learning is inherently heterogeneous.

Outliers and bias: To detect and lessen the impact of biased or malicious clients, mitigation techniques, such as detecting and suppressing potential outliers (DPSO), are used.

Non-IID data: While traditional aggregation methods such as Krum [2], Median [14], and Trimmed Mean show variable effectiveness in non-IID scenarios, they may not perform as well in complex datasets or with high adversary counts.

2.3. Defense mechanisms against malicious clients

Federated learning is vulnerable to an array of poisoning attacks, including:

Data poisoning: Adversaries may manipulate local datasets with the intent to undermine model performance [1] . Conventional validation-based techniques prove inadequate in federated learning due to the absence of a centralized validation data set [10].

Model poisoning: Malicious clients transmit altered updates to the server. Defense strategies such as Krum and Median aggregation provide resilient safeguards; however, they are predicated on the assumption of IID data, which is frequently unrealistic within distributed contexts.

In order to address potential outliers, score-based and vote-based weight assignment methodologies have been proposed. These strategies employ similarity metrics, such as cosine similarity, to filter updates prior to the aggregation process. In the

present study we put forth an innovative server-side filtering paradigm that systematically eliminates erroneous model parameters prior to the aggregation process. In contrast to conventional approaches such as Krum or Median, our methodology demonstrates computational efficiency and efficacy under non-IID frameworks. Through the filtration of malicious and outlier parameters, the proposed technique significantly augments global-model accuracy while concurrently safeguarding the confidentiality of local data.

3. Methodology

This section consists of the overview and framework for federated learning.

Summary of the suggested method: The approach presented in Figure 1 describes an advanced server-side filtering system that is integrated into a federated learning framework and is intended to improve model accuracy while simultaneously addressing data privacy and security concerns. The server creates a global model at the start of the process and distributes its parameters to a randomly selected subset of clients. Using its own data set and the parameters it has received, each client trains the model locally before sending the modified parameters back to the server. Before aggregating parameters meant for the global model's improvement, the server uses a filtering technique to remove any updates deemed malicious or unusual.

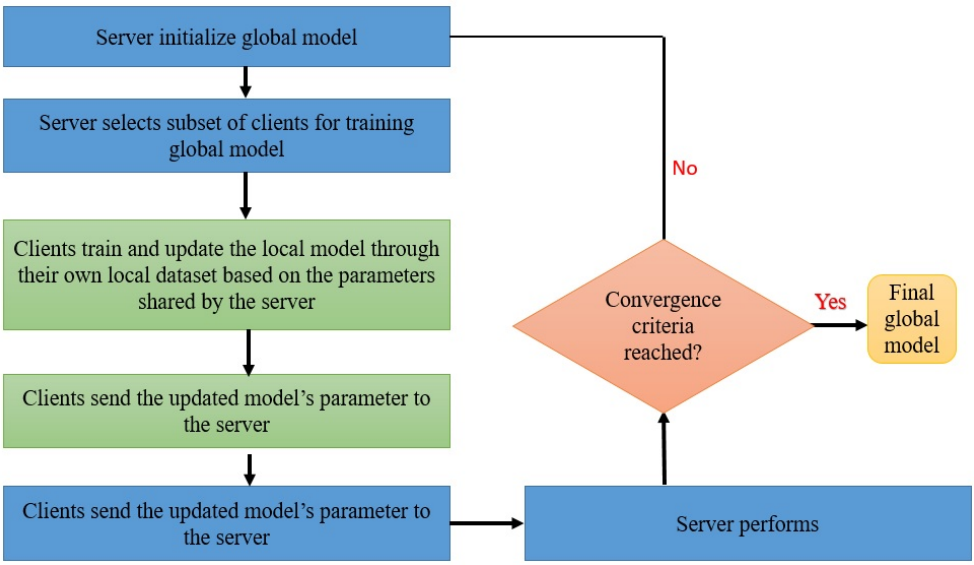

Figure 1. Workflow of our proposed approach

The federated learning framework (Figure 2) shows the fundamental architecture of federated learning. The K clients collaborate to enhance a global model while each

keeps a unique local data set represented by D_k. A subset of clients receive the model parameters after the server initializes them. The clients then use their own data to train their local models, modifying the parameters according to predetermined batch sizes and epochs. The modified parameters are then sent to the server, which aggregates them to allow the global model to be updated. This sequence represents one communication round represented by this sequence; in order to obtain the required degree of accuracy. several rounds of communication are conducted.

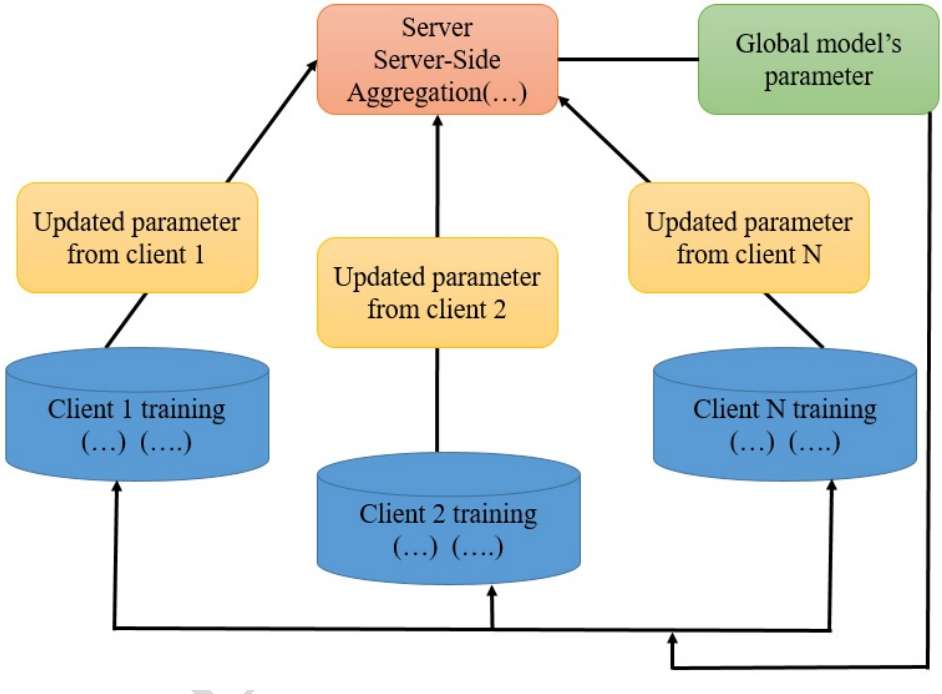

Figure 2. Basic framework of federated learning

The optimization objective for federated learning is:

$$\min_{w \in \mathbb{R}^d} \frac{1}{K} \sum_{k=1}^{K} \frac{n_k}{n} L_k(D_k, w) \tag{1}$$

where $n_k = |D_k|$ signifies the size of the data set associated with the k^{th} client and $n = \sum_{k=1}^{K} n_k$ represents the size of the cumulative data sets across all clients. Federated learning has numerous limitations and difficulties:

Non-IID data training dynamics: These are distorted by client datasets that often lack representations of the global population.

Resource restrictions: Batch sizes and training epochs may be limited by clients' limited processing capacity.

Client 'availability: Due to competing user applications or resource constraints, not all clients are always available for training.

Malicious clients: Adversarial actors may increase communication overhead and jeopardize global model performance by introducing tainted data or altered updates.

The proposed server-side filtering mechanism effectively addresses these challenges by discerning and excluding anomalous updates prior to aggregation, thereby enhancing model robustness and diminishing susceptibility to malicious activities.

3.1. Proposed algorithm

This algorithm is meticulously crafted to exclude potentially fallacious updates originating from malevolent clients within a federated learning framework. The server computes a centroid utilizing the median of client updates and assesses the cosine distance to discern updates that are in closest proximity to the centroid. The subset of updates nearest to the centroid is synthesized to refine the global model.

In Figure 2, the fundamental structure of federated learning is displayed. It is evident from earlier talks that the model's performance is negatively impacted when incorrect parameters are included in server-side aggregation. Malicious clients, according to our paper, are those that possess skewed outlier data and will provide the server with incorrect model parameters. A server-side filter is what we have suggested. Before the parameters are aggregated the filter is always run. We now give a detailed description of our methodology.

Centroid: Suppose we have m parameters, i.e., weights and bias. At time t, the i-th parameter of k-th client is denoted by $\Delta w_t^{k,i}$. Suppose the server receives the update from u such clients at time t. The median of the i-th parameter, med_t^i, is the median of the set $\{\Delta w_t^{1,i}, \Delta w_t^{2,i}, \ldots, \Delta w_t^{k,i}\}$. Finally, the centroid at time t is represented by vector $\mathbf{c}_t$, where it is represented by the following equation:

$$\mathbf{c}_t = [\text{med}_t^1, \text{med}_t^2, \ldots, \text{med}_t^m] \tag{2}$$

The reason behind the selection of the median is that our approach considers the possibility of malicious-outlier-weight updates sent to the server. The median as a representative of $\{\Delta w_t^{1,i}, \Delta w_t^{2,i}, \ldots, \Delta w_t^{k,i}\}$ is more robust than considering the mean $\{\Delta w_t^{1,i}, \Delta w_t^{2,i}, \ldots, \Delta w_t^{k,i}\}$.

Cosine distance: Suppose at time t the parameter update received from the k-th client is denoted by an m-dimensional vector Δw_t^k represented by:

$$\Delta w_t^k = [\Delta w_t^{k,1}, \Delta w_t^{k,2}, \ldots, \Delta w_t^{k,m}] \tag{3}$$

The cosine similarity [6] between the $\mathbf{c}_t$ and Δw_t^k, i.e., $\text{CoS}(\mathbf{c}_t, \Delta w_t^k)$ is calculated by the following equation:

$$\text{CoS}(\mathbf{c}_t, \Delta w_t^k) = \frac{\mathbf{c}_t \cdot \Delta w_t^k}{\|\mathbf{c}_t\|_2 \|\Delta w_t^k\|_2} \tag{4}$$

The rationale for opting for cosine distance instead of alternative metrics is that cosine distance shines as the superior choice when the emphasis lies on the orientation rather than the size of the vectors, rendering it perfect for intricate, high-dimensional datasets. In contrast to Euclidean distance, which reacts strongly to variations in scale and absolute values, cosine distance gauges the angle formed between vectors, ensuring that analogous patterns are detected regardless of the vectors' lengths. This characteristic fortifies its resilience against anomalies. Moreover, cosine distance alleviates the challenges posed by the curse of dimensionality, making it a more potent tool for navigating high-dimensional data, when Euclidean distance often falters in delivering significant distinctions.

To measure the similarity between the centroid c_t and a parameter update Δw_t^k, we utilize the Euclidean norm ($\|\cdot\|_2$) . The cosine distance between c_t and Δw_t^k, denoted as $\text{CoDis}(c_t, \Delta w_t^k)$, is computed using the following equation:

$$\text{CoDis}(c_t, \Delta w_t^k) = 1 - \text{CoS}(c_t, \Delta w_t^k), \tag{5}$$

where $\text{CoS}(c_t, \Delta w_t^k)$ represents the cosine similarity between c_t and Δw_t^k.

For u clients that send updates, pairwise distances from c_t to all received parameter updates are calculated, i.e., $\text{CoDis}(c_t, \Delta w_t^1), \text{CoDis}(c_t, \Delta w_t^2), \ldots, \text{CoDis}(c_t, \Delta w_t^u)$.

Algorithm 1 Server Execution Process

Set initial model parameters: w_0
for each communication round $t = 1, 2, \ldots$ **do**
  Randomly sample u_t clients for the current round.
  for each selected client $k \in u_t$ **in parallel do**
    Update client parameters: $w_t^{k+1} \leftarrow \text{ClientUpdate}(k, w_t)$
  end for
  Compute centroid c_{t+1} as per Equation (2).
  for each client $k \in u_t$ **do**
    Calculate distance: $\text{CoDis}(c_{t+1}, w_t^{k+1})$.
  end for
  Select the $\frac{u_t}{2}$ closest w_t^{k+1} to c_{t+1}.
  Compute total weight $n = \sum_{k=1}^{u_t/2} n_k$.
  Aggregate the updates:

$$w_{t+1} = \frac{\sum_{k=1}^{u_t/2} n_k \cdot w_t^{k+1}}{\sum_{k=1}^{u_t/2} n_k}.$$

end for

Algorithm 2 Client Update (k, w_t)

1: Split the data D_k into b batches.
2: **for** each iteration $i = 1$ to b **do**
3: Update weights:

$$w_t \leftarrow w_t - \eta \nabla L_k(D_k, w_t).$$

4: **end for**
5: return w_t to the server.

The proposed algorithm operates as follows: First, the centroid c_t is computed using the parameter updates received from all clients. Next, the cosine distance from the centroid to each parameter update is calculated. Subsequently, the parameters of the closest $u/2$ clients to the centroid are selected, where u is the total number of clients providing updates. These $u/2$ clients' parameters are considered for aggregation at the server side.

This selection strategy accounts for potential errors in client updates. Under ideal conditions, none of the selected clients provide erroneous updates, while in the worst-case scenario all selected clients may contribute erroneous updates. The algorithm assumes that, on average, up to half of the clients may send erroneous updates, thereby selecting the best $u/2$ parameters to minimize the impact of such errors. The detailed steps are outlined in Algorithm 1.

4. Experimental results

This section is composed of an overview of the dataset, performance metrics, experimental setup, and discussion of results.

4.1. Data set

The data set encompasses the execution durations for the multiplication of two 2048×2048 matrices utilizing a GPU OpenCL SGEMM kernel. It comprises 14 parameters: the initial 10 are ordinal and are constrained to values that correspond to four distinct powers of two, while the remaining four parameters are binary in nature. Outcomes from four executions for each parameter combination are incorporated into the data set and are documented in milliseconds.

Significantly, kernel limitations restrict the aggregate number of parameter combinations to 241,600 from 1,327,104 potential combinations. As elucidated in [4], this data set is an invaluable resource for machine-learning-driven auto-tuning, which is intended to enhance the performance portability of OpenCL applications.

Table 1 provides detailed descriptions of each parameter. The data set has no missing values or outliers, ensuring its reliability for experimental analysis.

The data set includes four outputs for each case, named Run1, Run2, Run3, and Run4. These outputs range between 13.25 and 3397.08 ms. For our analysis, Run1 is considered the dependent parameter.

Table 1
Description of independent parameters

Parameter	Feature type	Description
MGW	Int	MWG (matrix width global) defines the width of 2D tiles in matrix operations at the work group level. Possible values: 16, 32, 64, 128.
NWG	Int	NWG (number of work groups) defines the height of 2D tiles in matrix operations. Possible values: 16, 32, 64, 128.
KWG	Int	KWG (kernel work group) specifies the number of threads in the x-dimension of a work group. Possible values: 16, 32.
MDIMC, NDIMC	Int	Sizes of the workgroup in the x and y dimensions. Possible values: 8, 16, 32.
MDIMA, NDIMB	Int	Tiling dimensions within the kernel. Possible values: 8, 16, 32.
KWI	Int	Kernel loop unrolling factor. Possible values: 2, 8.
VWM, VWN	Int	Per-matrix vector widths for loading and storing. Possible values: 1, 2, 4, 8.
STRM, STRN	binary	Determines whether a thread accesses off-chip memory in a strided manner. Possible values: 0, 1.
SA, SB	binary	Enables caching of matrices A and B within the tile during computation. Possible values: 0, 1.

4.2. Experimental setup

The experiment was conducted using Google Colab, a cloud-based Jupyter notebook environment. A total of 400 clients were used, with the data set distributed unevenly among them. Clients run Minibatch SGD with a specific number of iterations (i) and batch size (b) on their local datasets. Malicious clients are simulated using a Gaussian distribution $N(0, 3)$.

4.3. Experimental results and discussion

The experiment examines the impact of malicious clients on the convergence rate and model performance in a federated learning setup for multiple linear regression. Experiments were conducted under three scenarios:

4.4. Base Case

In the first scenario, the federated version of multiple linear regression (MLR) is implemented without malicious clients. A total of 400 clients participate, running Minibatch SGD with $i = 3$ and $b = 10\%$ of their local data set, with a learning rate of 0.01.

Table 2 summarizes the results. To introduce heterogeneity, data is distributed unevenly across clients. A portion of the data set is retained at the server as test data to evaluate the global model.

Table 2
Federated implementation of MLR without malicious clients

Clients	Cycles	Speedup	MSE
5	666	1	1438.05
50	68	9.79	1389.08
100	34	19.58	1373.16
200	17	39.17	1454.70
300	12	55.50	1187.99
400	9	74.00	1155.12

The experiment starts with five clients, in which the server communicates with only these clients in each round. After each communication round the server aggregates model parameters to construct the global model and evaluates its performance using MSE on the test data. The model converged after 666 cycles with five clients. Increasing the number of clients to 50 significantly reduces cycles to 68, achieving a speedup of approximately $666/68 \approx 9.79$ and an MSE of 1389.08. Results indicate that increasing the number of clients per round accelerates convergence.

5. Impact of malicious clients on federated learning

The experiments are divided into two cases: (1) increasing the number of clients to enhance parallelism, and (2) increasing the computational effort at the client side by adjusting the batch size and number of iterations.

5.1. Increasing parallelism

In this scenario, client-side computation is limited to ensure consistent computational load across clients. Each client executes mini-batch stochastic gradient descent (SGD) with a fixed number of iterations ($i = 3$) and a batch size (b) equivalent to 10% of its local data set. The learning rate is set to 0.01.

Table 3 summarizes the experimental results, showcasing the effectiveness of the proposed server-side filtering technique compared to the traditional median filter and the absence of any filter. For this experiment, we assume that 10% of the selected

clients at each communication round are malicious. The experiment is performed with varying numbers of clients $(5, 50, 100, 200, 300, 400)$.

According to Table 4, the number of transmission cycles is fixed for every case to guarantee a fair comparison. For instance, in the case of five clients, the number of cycles is fixed at 666. For each client configuration the model's accuracy is recorded under three scenarios: (1) no server-side filter, (2) a median filter applied at the server, and (3) the proposed novel filter applied at the server.

The findings show that the suggested filtering method continuously performs better than both the conventional median filter or no filter at all. This demonstrates the unique filter's resilience and efficacy in reducing the negative effects of malicious updates on model performance.

Table 3
Effectiveness of proposed novel 'filter under increased parallelism

Clients	Cycles	MSE (No Filter)	MSE (Median Filter)	MSE (Novel Filter)
5	666	42362.70	114064.40	111726.70
50	68	103472.01	67790.20	60999.98
100	34	104051.91	48158.56	41998.24
200	17	102087.72	31601.58	27581.56
300	12	107364.54	25154.58	22854.02
400	9	110750.52	22222.27	20652.23

Table 4
Effectiveness of proposed novel filter under increased client-side computation

Local Batch Size [%]	MSE (No Filter)	MSE	MSE (Novel Filter)
10	104049.81	48187.96	41998.24
15	159222.2	53156.62	47327.51
20	107818.7	48727.99	41961.31
50	106240.6	49326.41	42919.67

5.1.1. Key Observations

1. In comparison to the median filter and the no-filter scenario, the proposed filter exhibits better model accuracy in every configuration.
2. As the number of customers increases the system becomes more parallel, reducing the number of cycles required for convergence.
3. The outcomes confirm that even under challenging circumstances the suggested method is efficient at managing harmful client updates in federated learning configurations.

5.2. Increasing computation per client

There is a cap on the total number of clients. The results of the federated implementation of multiple linear regression without malicious clients are shown in Table 2. The total number of clients is fixed at 10. One hundred clients are chosen for each communication round, with three iterations. The batch size varies between 10%, 15%, 20%, and 50% of the local data set for each client.

As observed in Table 3, the global model converges after 34 communication rounds under the given conditions. To evaluate performance, the experiment was conducted under three different circumstances: (1) no server-side filter, (2) a median filter applied at the server, and (3) the proposed novel filter applied at the server. The results show that the proposed filter consistently improves the model's performance compared to the traditional median filter and the absence of any filter, highlighting its robustness and effectiveness in mitigating the impact of malicious updates.

5.2.1. Key observations

1. Increasing the batch size improves the computational load per client, resulting in faster convergence and improved performance.
2. The proposed filter outperforms both the median filter and the absence of any filter, delivering consistently lower MSE.
3. Fixing the number of communication rounds (34) ensures a fair comparison across all scenarios.

Figures 3 and 4 present a comparative analysis highlighting the enhanced performance of the proposed filtering technique compared to the median-based filtering approach. The results demonstrate its superiority in both scenarios, with increased levels of parallelism and a higher number of client-side computations.

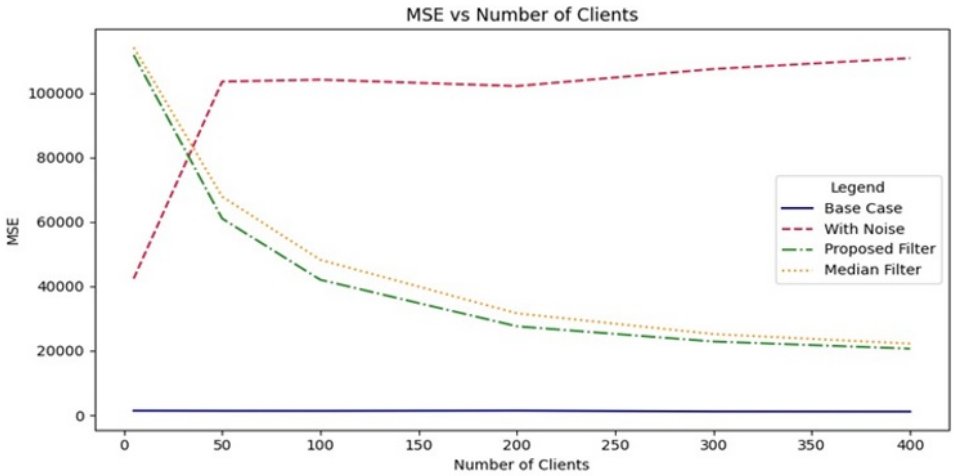

Figure 3. Comparison of our proposed filter vs. median filter under increased parallelism

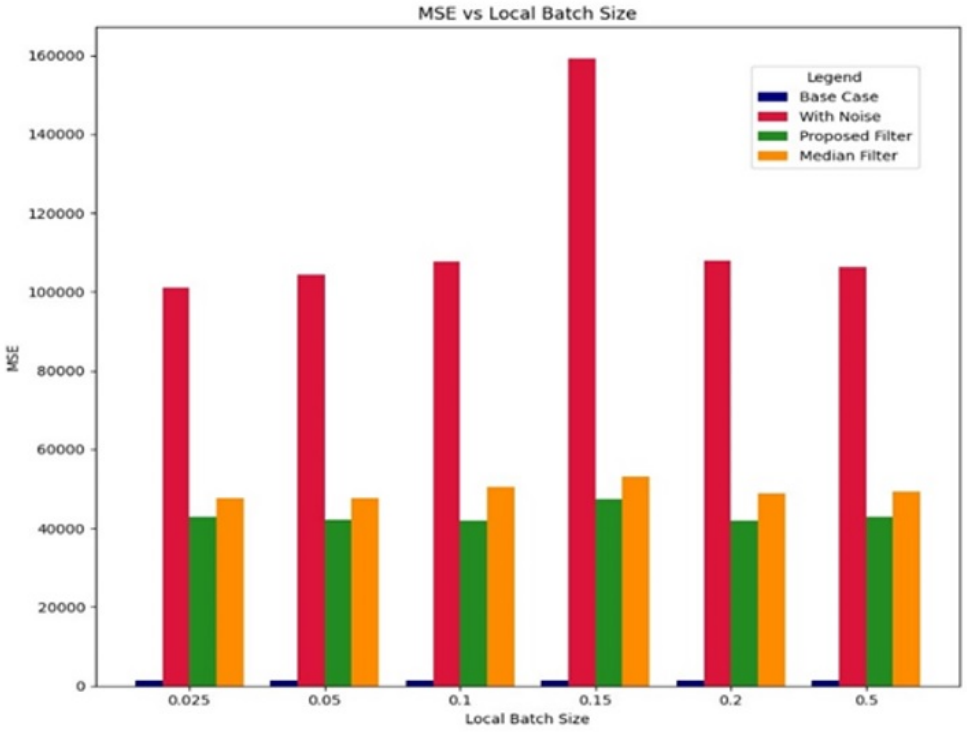

Figure 4. Comparison of our proposed filter vs. median filter under client-side computations

5.2.2. Complexity analysis of algorithm

The time complexity to compute c_t is $\mathcal{O}(mk)$, where m represents the dimensionality of the updates received from k clients (lines 1–6). Computing the cosine distance between c_t and the k-th client update, Δw_t^k, has a complexity of $\mathcal{O}(m)$. If there are u such clients, the complexity of calculating the cosine distance from c_t to each update becomes $\mathcal{O}(mu)$ (lines 7–8).

The next step involves selecting the closest $u/2$ clients to c_t, which has a complexity of $\mathcal{O}(u \log u)$. Finally, the complexity of computing the average is $\mathcal{O}(mu)$. Therefore, the overall complexity of the algorithm is:

$$\mathcal{O}(mk + mu + u \log u) \approx \mathcal{O}(mk)$$

5.2.3. Complexity analysis of algorithm: client update

The data set D_k, owned by client k is divided into b batches, resulting in $\frac{D_k}{b}$ updates. The complexity of each update is approximately $\mathcal{O}(|b|)$, where $|b|$ is the batch size.

Therefore, the overall complexity for a single client is:

$$\mathcal{O}\left(\frac{D_k}{b} \cdot |b|\right)$$

Since all clients perform the update operation in parallel, the overall complexity for a single communication round remains:

$$\mathcal{O}\left(\frac{D_k}{b} \cdot |b|\right)$$

6. Conclusion and future work

In this manuscript we introduced an innovative filtering method specifically formulated to tackle the complexities associated with malicious updates in the context of federated learning. Our methodology presents a resilient server-side filtration mechanism that demonstrably surpasses traditional median-based filtering approaches. The filter we propose facilitates expedited convergence and enhanced model accuracy, even with a constrained number of communication iterations, thereby illustrating its efficacy and practical relevance in federated learning frameworks.

The findings of our investigation underscore the merits of the proposed technique in contexts where robust and effective filtering is paramount. By diminishing the impact of malicious or anomalous updates, the proposed filter not only bolsters the reliability of the training regimen but also exhibits significant enhancements in the overall efficacy of the federated learning architecture.

Notwithstanding its encouraging results, the present research is subject to specific limitations that necessitate further inquiry. A principal limitation pertains to the presumption of IID data among clients. While this presumption simplifies the analytical challenge, actual federated learning situations frequently encompass non-IID data distributions, wherein client datasets can differ markedly due to diverse local environments and usage patterns. Subsequent research endeavors will concentrate on adapting the proposed filtering framework to accommodate non-IID data distributions, thereby augmenting its pertinence and utility.

Another constraint resides in the scale of the experiments undertaken. The current investigation predominantly addresses regression tasks with a relatively limited number of model parameters. To tackle more intricate challenges, such as classification and deep learning applications, we intend to incorporate deep-learning-centric methodologies into our framework. These methodologies will facilitate the effective management of larger and more complex models, thereby extending the applicability of the proposed filter to a broader array of demanding federated learning issues.

Moreover, forthcoming research will encompass the assessment of the proposed filter across a variety of datasets and settings to enhance our comprehension of its

performance under disparate conditions. This will include the evaluation of the technique with heterogeneous client systems, diverse data distributions, and varied model architectures.

In summary, although the proposed filter signifies substantial progress in federated learning by improving robustness and performance, further scholarly work is required to rectify its existing limitations and broaden its applicability. By integrating these enhancements, the proposed approach possesses the potential to serve as a pivotal facilitator of dependable and efficient federated learning systems in practical applications. This expanded version thoroughly elucidates the contributions, limitations, and prospective avenues of inquiry while upholding a professional and cohesive framework.

Author contribution

Abhijit Debnath: supervision, investigation, methodology, analysis, software, review.
Uddalok Sen: conceptualization, solution methodology, data creation, formal analysis, software and editing.
Uddalak Seal: conceptualization, solution methodology, data creation, formal analysis, software;
Abhoy Sarkar: conceptualization, solution methodology, data creation, formal analysis, software.

Acknowledgements

We would like to express our sincere gratitude to the administration of IEM, Newtown, UEM, Kolkata, as well as the leadership of the MCKV Institute of Engineering, Liluah, Howrah, for its unwavering support in research and development initiatives.

References

[1] Biggio B., Nelson B., Laskov P.: Poisoning attacks against support vector machines, *arXiv preprint arXiv:12066389*, 2012.

[2] Blanchard P., El Mhamdi E.M., Guerraoui R., Stainer J.: Machine learning with adversaries: Byzantine tolerant gradient descent, *Advances in neural information processing systems*, vol. 30, 2017.

[3] Bonawitz K., Ivanov V., Kreuter B., Marcedone A., McMahan H.B., Patel S., Ramage D., Segal A., Seth K.: Practical secure aggregation for privacy-preserving machine learning. In: *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017. doi: 10.1145/3133956.3133982.

[4] Falch T.L., Elster A.C.: Machine learning based auto-tuning for enhanced opencl performance portability. In: *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pp. 1231–1240, IEEE, 2015. doi: 10.1109/ipdpsw.2015.85.

[5] Geyer R.C., Klein T., Nabi M.: Differentially private federated learning: A client level perspective, *arXiv preprint arXiv:171207557*, 2017.

[6] Lahitani A.R., Permanasari A.E., Setiawan N.A.: Cosine similarity to determine similarity measure: Study case in online essay assessment. In: *2016 4th International conference on cyber and IT service management*, pp. 1–6, IEEE, 2016. doi: 10.1109/citsm.2016.7577578.

[7] Liu J., Huang J., Zhou Y., Li X., Ji S., Xiong H., Dou D.: From distributed machine learning to federated learning: A survey, *Knowledge and Information Systems*, vol. 64(4), pp. 885–917, 2022. doi: 10.1007/s10115-022-01664-x.

[8] McMahan B., Moore E., Ramage D., Hampson S., y Arcas B.A.: Communication-efficient learning of deep networks from decentralized data. In: *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[9] Rivest R.L., Adleman L., Dertouzos M.L., *et al.*: On data banks and privacy homomorphisms, *Foundations of secure computation*, vol. 4(11), pp. 169–180, 1978.

[10] Steinhardt J., Koh P.W.W., Liang P.S.: Certified defenses for data poisoning attacks, *Advances in neural information processing systems*, vol. 30, 2017.

[11] Tian Y., Zhang W., Simpson A., Liu Y., Jiang Z.L.: Defending against data poisoning attacks: from distributed learning to federated learning, *The Computer Journal*, vol. 66(3), pp. 711–726, 2021. doi: 10.1093/comjnl/bxab192.

[12] Wang L., Meng Z., Yang L.: A multi-layer two-dimensional convolutional neural network for sentiment analysis, *International Journal of Bio-Inspired Computation*, vol. 19(2), pp. 97–107, 2022. doi: 10.1504/ijbic.2022.121236.

[13] Yang Q., Liu Y., Chen T., Tong Y.: Federated machine learning: Concept and applications, *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10(2), pp. 1–19, 2019. doi: 10.1145/3298981.

[14] Yin D., Chen Y., Kannan R., Bartlett P.: Byzantine-robust distributed learning: Towards optimal statistical rates. In: *International conference on machine learning*, pp. 5650–5659, Pmlr, 2018.

[15] Zhang X., Fu A., Wang H., Zhou C., Chen Z.: A privacy-preserving and verifiable federated learning scheme. In: *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2020. doi: 10.1109/icc40277.2020.9148628.

Affiliations

Abhijit Debnath
University of Engineering and Management, Department of Computer Science & Engineering, Institute of Engineering and Management, Newtown, Kolkata, India, 700160, abhijit.dbnath@gmail.com; abhijit.debnath@uem.edu.in

Uddalok Sen
MCKV Institute of Engineering, Department of Computer Science & Engineering, Liluah, Howrah, India, uddalok.sen@mckv.edu.in

Uddalak Seal
University of Engineering and Management, Department of Computer Science & Engineering, Institute of Engineering and Management, Newtown, Kolkata, Inida, 700160,
uddalak.seal@iem.edu.in

Abhoy Sarkar
University of Engineering and Management, Department of Computer Science & Engineering, Institute of Engineering and Management, Newtown, Kolkata, India, 700160,
abhoy.sarkar@uem.edu.in

Received: 22.02.2025
Revised: 11.04.2025
Accepted: 15.08.2025