ORIGINAL ARTICLE

# Robust Federated Learning against Poisoning Attacks using Capsule Neural Networks

**Mohsen Sorkhpour**[1] | **Mohsen Rezvani**[2] | **Esmaeel Tahanian**[3] | **Mansoor Fateh**[4]

[1, 2, 3, 4] Faculty of Computer Engineering, Shahrood University of Technology, Shahrood, Iran

**Correspondence**
Mohsen Rezvani, Faculty of Computer Engineering, Shahrood University of Technology, Shahrood, Iran
Email: mrezvani@shahroodut.ac.ir

The expansion of machine learning applications across different domains has given rise to a growing interest in tapping into the vast reserves of data that are being generated by edge devices. To preserve data privacy, federated learning was developed as a collaboratively decentralized privacy-preserving technology to overcome the challenges of data silos and data sensibility. This technology faces certain limitations due to the limited network connectivity of mobile devices and malicious attackers. In addition, data samples across all devices are typically not independent and identically distributed (non-IID), which presents additional challenges to achieving convergence in fewer communication rounds. In this paper, we have simulated attacks, namely Byzantine, label flipping, and noisy data attacks, besides non-IID data. We proposed Robust federated learning against poisoning attacks (RFCaps) to increase safety and accelerate convergence. RFCaps is a prediction-based clustering and a gradient quality evaluation method to prevent attackers from the aggregation phase by applying multiple filters and also accelerate convergence by using the highest quality gradients. In comparison to MKRUM, COMED, TMean, and FedAvg algorithms, RFCap has high robustness in the presence of attackers and has achieved a

higher accuracy of up to 80% on both MNIST and Fashion-MNIST datasets. **Keywords** — Federated Learning, Capsule Neural Networks, Poisoning Attacks, Clustering.

## 1 | INTRODUCTION

smartphones become more prevalent, a huge amount of data has been generated. Many companies have been attracted to the vast amount of data stored in these devices, however, exploitation of such a data poses privacy concern. To realize this idea with respect to privacy-preserving, federated learning has been proposed. Federal learning is a modern distributed learning method, which is provided to meet some of the most important and vital need in the field of machine learning, especially neural networks and deep learning, which is the existence of a great deal of training data. It is extremely valuable to have access to a large amount of data without violating the privacy of individuals.

Federated Learning seeks to learn a global model by conducting distributed training on participating devices and integrating local models into a global model. Privacy preserving is not the only advantage of federated learning. This method can be used as a mechanism to cope with the increasing complexity of machine learning models and the amount of data available to train them, reduce hardware costs for storing and processing information, reduce costs associated with securing individuals' private data, speed up training time, reduce the risk of leaking confidential information, and increase the accuracy of federated trained models which are other benefits of using a federated learning technique [11].

Recent research has shown; however, this type of learning method has many vulnerabilities since the parties cannot be totally controlled [8]. These vulnerabilities can reduce the accuracy of the learner model or change its behaviour, result in the leak of information that we intend to protect their privacy by learning in a federated way. Because most of the implementations have been made to exploit data generated by mobile phones, another very important challenge of federated learning is the limited processing power of these devices compared to powerful servers, which prevents the use of complex models with a large number of learning parameters such as capsule neural network [11]. In this way, these models cannot be utilized to their full potential. Nevertheless, it is always important to keep in mind that models should be designed and used as efficiently as possible and with the least number of parameters.

Furthermore, since the majority of platforms for use the federated learning method are based on the Internet, and due to its infrastructure shortcomings, such as limited communication bandwidth and significant communication latency [6], we are facing to another challenge, which is called communication overhead. Communications overhead is a critical bottleneck in federated learning, and with the growth of the number of participates, communications overhead could be increased explosively [25]. Therefore, the value of exchanged gradient over the network and the number of required communication rounds must be considered in order to reduce the communication overhead.

We proposed a comprehensive solution that addresses the above challenges by considering all aspects. First, by minimizing the number of parameters of the capsule neural network, we made it able to be trained using mobile processors. Thus, by reducing the learning parameters, the size of the model has been significantly reduced. In addition, its transmission along the network has been facilitated, which will greatly help the communication overhead. Using DigitCaps layer interpretability, we have also developed a very precise clustering procedure to reduce the biases caused by non-IID data in local datasets, which greatly accelerated convergence. Last but not least, to mitigate the

presence of attackers during the aggregation process, we have measured the accuracy of each model and identified the malicious or useless models. As a result, we have seen a high degree of resistance, we conduct extensive experiments and evaluate our proposed method's efficiency and robustly using two famous datasets in different settings. To recapitulate, in this implementation, much effort has been done to make federated learning a fast and secure method for use in various applications. In summary we make the following contributions:

- Adding the trainability of complex networks to the federated learning method by reducing the learning parameters in an experimental way, in order to reduce the computational and communication overhead.
- Proposing a precise clustering method in order to reduce biases and accelerate the convergence.
- proposing a robust aggregation method by applying multiple filters to hinder attackers from participating in the aggregation process.

The rest of this paper is organized as follows; Section 2 presents the related works. Section 3 describes the basic concepts of federated learning and its threats, and also in this section, capsule neural networks has been introduced. Section 4 present our proposed method. Section 5 shows our experiment and results. Section 6 presents the discussion. Finally, the paper is included in section 7.

## 2 | RELATED WORKS

In recent years, many studies have been conducted to address the challenges of federated learning by introducing a variety of aggregation methods. However, the majority of them focused on only one challenge. FedAvg [13] updates the global model by averaging the gradients of the local models. This algorithm was used for recognizing out-of-vocabulary words [17] and also improving the mobile keyboard prediction [18]. Federated Stochastic Variance Reduced Gradient (FSVRG) [9] was proposed to improve the efficiency of FedAvg by dealing with sparse data. In [19] CO-OP algorithm has been proposed for asynchronous model updates, in contrast to FedAvg and FSVRG, it merges any received client gradients with the global model. According to the difference in the age of the models, the local and global models merging is carried out using a weighting scheme, instead of directly averaging the models.

Due to the distributed scheme of federated learning, it is highly vulnerable to attacks against the learning models. The most serious concern for federated learning is faulty or malicious updates sent by clients. As a result, standard federated learning algorithms such as FedAvg [20, 21] are vulnerable to both model poisoning and data poisoning. Researchers have proposed different robust averaging algorithms to overcome this problem [15, 14, 23].

Some other researches have focused on vulnerability in federated learning known as a backdoor attack [23, 24]. In this type of attack, the adversary attempts to reduce the model's performance on targeted tasks while keeping the main tasks' performance acceptable [24].

Authors in [1] proposed a reinforcement agent to select the most optimal clients in each aggregation, instead of selecting a random subset of participants only to accelerate convergence. In this method, a large computational overhead is imposed on the coordinating server to train the reinforcement learning agent. The training of a reinforcing agent is a long process, not a solution to certain problems where speed is crucial. Jiang et al [2] proposed partial aggregation, a method that tries to select a random subset of the weights of each model in each aggregation round instead of using the total weights of each model to make the aggregation process faster due to reduced calculations. In this method, no attackers were assumed and no evaluation is done on the clients.

Muñoz-González et al [3] Strong Byzantine federated learning by averaging the adaptive model, in this method, during successive server-side aggregation processes, it identifies and blocks each participant each time, which reduces the accuracy of the global model and then performs real aggregation without malicious participants. In this algorithm, the global model is not sent to malicious participants and remains blocked forever. One of the disadvantages of this method is that in the early rounds of aggregation due to non-IID data, which make the local model gradients fully inaccurate, the proposed method may block a healthy participant.

Authors in [4] proposed two defense algorithms, label-based semi-supervised defense, and clustering-based semi-supervised defense against label-flipping in correcting labels being attacked. In this research, only label-flipping attacks have been considered and there is minimum capability for the attacker. In other words, it is assumed that the attacker does not have direct access to the models and only can flip the labels. But In the real world, the attackers have gone beyond these assumptions.

Authors in [15] presented a robust Byzantine aggregation algorithm called KRUM, which tries to avoid the aggregation of deviated weights by using the Euclidean similarity criterion between the weights sent by clients. To solve the KRUM slow convergence problem, a faster algorithm called MKRUM was introduced.

Yin et al. [14], proposed a coordinate-wise median (COMED), a byzantine-robust statistical learning algorithm with a focus on statistical optimality, in this algorithm, the aggregation operation is performed by calculating the median of gradients in each layer.

In this research, we address two crucial challenges in the context of federated learning: safety and convergence speed. Our approach centers on optimizing the learning process by minimizing the number of learning parameters and selecting updates of the highest quality. This enables us to incorporate more complex models in the federated learning framework, thereby enhancing convergence speed and safety significantly. To tackle the issue of non-IID data, we propose robust federated learning algorithms that effectively mitigate its impact. Robust federated learning against poisoning attacks (RFCaps), detects and rejects unreliable updates of participating clients while speeding up the convergence process. By incorporating these strategies, we aim to achieve superior performance and robustness in federated learning scenarios, ultimately enabling efficient training of complex models on non-IID data.

## 3 | BASIC CONCEPTS

As federated learning has become increasingly popular, different categories have been created for this kind of learning. To determine the application of the proposed method, we try to briefly describe these categories.

### 3.1 | Federated Learning Methods

Based on the distribution of data characteristics and data samples belonging to clients, the federated learning method can be classified into three groups [12]. These three groups are as follows:

- Horizontally Federated Learning (HFL)
- Vertical Federated Learning (VFL)
- Federated Transfer Learning (FTL)

In the category of horizontal federated learning or sample-based learning, clients have datasets with similar

features. But we see the variation in the sample data, for example, customer data for several online stores that have similar features but belong to different customers. In vertical federated learning or a feature-based approach, clients have the same data sample. In other words, the data belong to a specific identity (ID) while sharing different features. The customers of a bank and the purchase history of the same customers in an online store are some examples of this category. There are both feature differences and data samples in the federated transfer learning method. For example, data from a bank in one country and data from an online store in another country.

In the proposed method, we will focus on horizontal federated learning (HFL). In this type of federated learning, two datasets share the same sample ID space but differ in feature space. It should be noted that the horizontal federated learning method is also divided into two subsets H2B[1] and H2C [2] [8]. Under H2B, there are typically a small number of participants, and each participant can be selected repeatedly throughout the federal learning process. Participants also have considerable computing power and advanced technical capabilities. Under H2C, there are thousands or even millions of potential participants. In each round of aggregation process, only a subset of them is selected. Because their datasets are small, they are unlikely to frequently select a participant for federated training. Participants also generally have limited computing power and little technical ability.

It should be noted that one of the basic assumptions in learning neural networks is the independence and uniformity of data distribution in the training process [10]. However, in federated learning, due to user preferences, the data available on all devices is not independent and has the same statistical distribution. So, the data in the devices is not IID[3]. For example, we want to train a model of a neural network that can classify several classes, including cars, flowers, houses, etc. In optimal conditions, the training data should be given to the model randomly and with a uniform distribution so that the training process is performed in the most efficient possible way by observing the IID data; but in the federated learning method, users' preferences violate this basic assumption. As an example, a user is interested in car images, and 80% of the images on this participant's smartphone are car-related images.

Therefore, the bias generated by the weights of models trained with non-IID data poses challenges to federal training. The results of experiments performed in [1] show that the lack of statistical independence of device data can lead to a slowdown in convergence in the process of federated learning training phase.

## 3.2 | Poisoning attacks

Recent studies have shown that federated learning can be exposed to a wide range of attacks. There are two common types of attacks including poison attacks and inference attacks; however, we will only address poison attacks in the proposed method.

According to the attacker's objective, poisoning attacks fall into two categories: random attacks and targeted attacks [30]. A random attack attempts to reduce the accuracy of the global model, while a targeted attack attempts to force the global model to output a target label specified by the adversary. The detection of a targeted attack is usually more challenging than a random attack since the attacker has a particular goal. Poisoning attacks can be performed on either the clients' local dataset, which is called data poisoning attack, or directly on gradients of the model which is called model poisoning attack. Thus, as these two types of poisoned updates can be sourced from clients' devices, just having access to the coordination server is enough for attackers to perform both poisoning attacks. We will describe these categories of poisoning attacks in the next couple of sections.

[1] HFL to businesses (H2B)

[2] HFL to consumers (H2C)

[3] Independent and identically distributed

### 3.2.1 | Data poisoning

Generally, there are two categories of data poisoning attacks these categories including clean-label [26] and dirty-label [27], clean-label attacks assume that the adversary cannot change the label of any training data since data are certified as belonging to the right class. Therefore, it can only add new samples to the dataset. While the attacker's injected training examples are cleanly labelled it makes imperceptible changes to the sample to create a poison instance. Although, in dirty-label poisoning attack adversary can change the label of training data. In this attack the adversary tends to poison the global model by sending poisoned gradients derived from mislabelled data.

In this paper, label-flipping attack [28,29] is used as one of the most common dirty-label poisoning attack. In this attack, the attacker tries to change the model behaviour or reduce the accuracy of the final model by flipping the training sample labels. For example, in the proposed method to increase the destructive intensity of attacks, we have acted in such a way that the attackers can change the label of more than one class. In this case, the accuracy has significantly reduced compared to flipping just one label. Another data poisoning attack that is very challenging to detect is backdoor attack [27]. In this attack, the malicious client tends to manipulate training data by applying some modification or adding specific features to original training data based on its mission in order to make the model behave as intended by the attacker.

### 3.2.2 | Model poisoning

As Federated learning's participants have control over their local datasets and gradients of their local model, more various type of poisoning attacks has been proposed. Model poisoning attack [23] is one of the most effective in these categories of attacks. It's aimed at poisoning local model updates before sending them to the server, and the attacking participant executes its malicious targets in the form of gradients sent to the server. It should be noted that the intensity of the impact of this type of attack is greater and more destructive than data poisoning.

In order to determine the robustness of the proposed method, along with other attacks we used, the byzantine attack [15,14] has been employed as a well-known model poisoning attack in the federated learning context.

## 3.3 | Capsule Neural Networks

This part of the article discusses the capsule neural networks. The main components of capsule neural networks include the definition of capsules, the dynamic routing algorithm, and their functionalities have been introduced in the next sections. Additionally, we have discussed why the capsule neural network was created, its advantages over traditional convolutional networks, and its weaknesses.

### 3.3.1 | Capsule Neural Network Overview

Capsule neural networks [5] are another type of neural network that was introduced in 2017 by Professor Hinton and colleagues as a response to some of the problems of convolutional neural networks. This network is also known as a modern neural network since the function of neurons in this network has changed from a scalar to a vector dimension. CNN networks have revolutionized machine learning, especially in the field of computer vision. Since the introduction of CNN networks, a lot of research has been done to improve the performance of these networks. This has resulted in the development of several architectures for CNN networks. One of the major problems with CNN networks is their inability to maintain the spatial relationships between the image elements at the training phase, therefore, CNN

networks are unable to recognize images that rotated at different angles [5] when applied.

Hence, a lot of efforts have been done to overcome the problem of detecting rotated images in CNN's networks. Data augmentation technique is one of the most widely used techniques to address this problem. It is a bunch of some classic image processing activities such as padding, cropping, zooming, random rotating, vertical and horizontal flipping, etc which apply to images to increase the amount of training data. However, this method is not considered an adequate response to this problem and is a way to reject the question. In the end, it can be concluded that CNN neural networks are highly dependent on training data and only correctly recognize images based on their training dataset.

As the main difference between CNNs and Capsule Neural Networks, it should be noted that the output and input of each neuron in traditional neural networks is a scalar number, which is obtained by multiplying the input by the weight of the neuron and then applying the activation function, While in capsule neural networks another concept called capsule is used instead of the neuron. Each capsule is a vector consisting of several neurons; in other words, the input and output of each capsule is a vector. Each capsule consists of an 8D vector of neurons. The length of each vector indicates the presence of an object and the direction of the vector indicates the instantiation parameters (position, size, orientation, etc) of each object [5]. Nevertheless, it is clear that the capacity of each capsule to feature extracting is greater than a single neuron, we will explain the functionality of each capsule and their relationship in the next subsections.

### 3.3.2 | Dynamic routing algorithm

The dynamic routing process is the most important part of a capsule neural network. During this process, the relationships between capsules of each layer and the subsequent layers are determined. A lower-level capsule prefers to send its output to higher-level capsules whose activity vectors have a large scalar multiplication by predicting of a lower-level capsule; in fact, in the bottom layer capsule routing process, the output of the next layer capsules is estimated. It should be noted that, as mentioned, the vector length of each capsule indicates the probability of a feature in the image, which is obtained by applying a non-linear squashing activation function to each vector. The following formula represents the squash function.

$$V_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \cdot \frac{s_j}{\|s_j\|} \tag{1}$$

In the above formula, v is the output of capsule j and $s_j$ is the input of capsule V. This activation function tries to preserve the direction of the vector while reducing its length [5]. Worth noting, except for the first layer of the capsule neural network, which is obtained from the convolution process, the input of all capsules is obtained from the weighted output of the capsules of the lower layer.

$$S_j = \sum_i C_{i,j}\, \widehat{U}_{i,j} \quad , \quad \widehat{U}_{i,j} = W_{i,j}\, U_i \tag{2}$$

As indicated above, $u_i$ also refers to the i-th capsule and $c_{i,j}$ represents the pairing coefficients. Using these coefficients, it is determined that the output of each capsule in the lower layer (i) should pass to which capsule in the upper layer (j). Also, the sum of all these coefficients between one capsule in each layer and all the capsules of

| symbol | description |
|---|---|
| N | The number of clients |
| NC | The number of classes in entire dataset |
| $\omega_t$ | The global model sent to the clients |
| $G_t$ | The set of clients considered good by the server |
| $N_G$ | The size of $G_t$ |
| $AG_t$ | The average of $G_t$ |
| C | The dictionary of clustered client's gradients |
| $CNO_i$ | The cluster number of i-th client |
| $Cn_i$ | The clients of i-th cluster |
| $DG_i$ | The gradients of i-th clients DigitCaps Layer |
| K | The number of needed clients from each cluster |

**TABLE 1** THE NOTATION USED IN THIS PAPER.

the upper layer is equal to 1, which is modified by the routing algorithm in an iterative process based on agreement. Which is determined as follows during the dynamic routing iterative process.

$$\begin{aligned} C_{i,j} &= softmax(b_{i,j}) \\ &= \frac{\exp(b_{j,j})}{\sum_k exp(b_{j,k})} \end{aligned} \tag{3}$$

According to the above formula, $b_{i,j}$ are the log prior probabilities that indicate that each i must be paired with capsule j in the upper layer; in other words, capsule i sends its output to capsule j. These probabilities are updated as discriminative along with the other gradients. As mentioned, the routing algorithm is agreement-based, the amount of agreement being measured between the output of the top-layer capsules ($V_j$)) and the estimate $\hat{u}_{i,j}$ calculated by the lower-layer capsules ($u_i$). In fact, this agreement is measured by using the scaler multiplication of the output of the capsules in layer j with the estimate calculated by the capsules in layer i. The following formula shows how these probabilities are calculated (r represents the r-th round of the routing process).

$$b_{i,j}^{r} = b_{i,j}^{r-1} + \hat{u}_{i,j} \cdot V_j \tag{4}$$

## 4 | PROPOSED METHOD

In this section, we present a comprehensive overview of the key components that constitute our proposed method, shedding light on their intricate functionalities and interactions. Our approach is designed to enhance the robustness and efficacy of federated learning systems, particularly in the presence of malicious attackers and non-IID data distributions.

## 4.1 | Overcoming Training Complexity

Despite the many advantages of capsule neural networks, there are some drawbacks to the performance of these networks. The most negative aspect of these networks, is the time-consuming training phase, due to the dynamic routing process between the capsules. As the number of network parameters increases, the number of capsules will increase depending on the network architecture. Therfore, the dynamic routing process between the capsules takes more time. The naive capsule neural network has approximately 6 million and 810 thousand learning parameters, as a result of which 1152 8-D capsules are created in the primary capsule layer. Training this number of parameters as well as routing between this number of capsules with the next layer is a time-consuming process and requires powerful processors.

Federated learning offers the potential to expedite the training of intricate models like capsule neural networks by capitalizing on substantial datasets and distributed computational resources. Nonetheless, several limitations necessitate careful consideration when employing federated learning in practice. The applicability of leveraging capsule neural networks within most federated learning environments is hindered by the constrained processing capabilities of devices such as cell phones and edge devices. Furthermore, the inherent limitations of most federated learning environments, characterized by limited processing power, necessitate attentive monitoring of data exchange volumes between servers and clients in centralized setups or among clients in fully decentralized configurations. The effective communication between clients or between clients and servers, along with the volume of gradients exchanged through the network, poses another significant challenge in federated learning called communication bottleneck. In the Robust federated learning against poisoning attacks (RFCaps), with an experience-based method, by reducing the number and size of kernels, we have a 63-fold reduction in the number of network parameters compared to the primary capsule neural network. As a result of these modifications, the number of parameters has been reduced to 107K; the network size from 27 MB to 420 KB, and the number of capsules in the primary capsule layer from 1152 to 32 capsules.

The reduction in the number of learning parameters in RFCaps has demonstrated promising outcomes by facilitating faster training of local models and minimizing data exchange volumes between participants. Consequently, this enables the effective utilization of capsule neural networks in federated learning environments. It is noteworthy that despite the reduction in the number of parameters, the resultant decrease in the accuracy of the final model is negligible when juxtaposed with the considerable benefits offered by employing such networks. Table1 contains a summary of the notations used in this paper.

## 4.2 | Clustering-based Approaches for Non-IID Data in Federated Learning

In federated learning, data is distributed among multiple devices or clients, and the data distribution across these devices is often non-IID, meaning that each device has different data characteristics and class distributions. This non-IID data distribution can lead to biased convergence, slow learning, and suboptimal model performance. The clustering-based method aims to address these issues by grouping similar data and gradients, which can help create a more balanced and representative dataset for training the global model and performing a balanced aggregation. Clustering-based methods offer notable advantages in the context of federated learning. Firstly, they effectively address bias issues by generating balanced representations of diverse data characteristics, thereby mitigating the adverse effects of non-IID data. Secondly, these methods expedite the convergence process by effectively grouping similar gradients, leading to a more focused and efficient training regime for the global model. Lastly, the aggregation of clustered gradients enhances communication efficiency by significantly reducing the volume of data exchanged between clients

and the server during the federated learning process. These advantages collectively contribute to the improved performance and effectiveness of federated learning algorithms when dealing with non-IID data distributions.

The clustering-based algorithm consists of two main steps. First, each client conducts local training using its own non-IID data and calculates the gradients representing the local updates. These gradients are then grouped together based on their similarities using clustering algorithms like K-Center [1]. Second, the server aggregates the gradients from clients within each cluster separately. By doing so, the algorithm helps to reduce bias and accelerate the convergence speed of the global model in federated learning. Only by clustering and optimally selecting gradients for aggregation, we will have a significant improvement in convergence speed [1]. This is because clustering can help to uniformize the statistical distribution of gradients and due to this the convergence will accelerate.

## 4.3 | RFCaps: Robust Federated Learning Against Poisoning Attacks

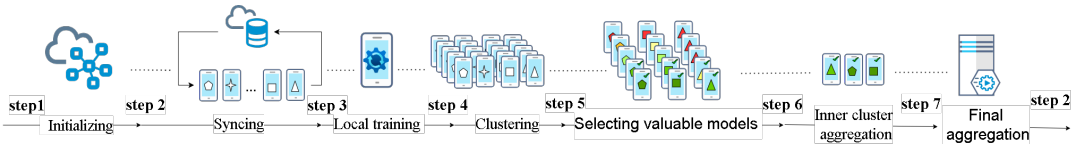


**FIGURE 1** The federated learning workflow with RFCaps.

In this study, we introduce a robust federated learning aggregation algorithm to effectively counter the challenges posed by poisoning attacks, termed RFCaps. Moreover, RFCaps adeptly addresses the bias introduced by non-IID data by leveraging a clustering-based approach. The RFCaps algorithm comprises two main stages: gradient clustering, which groups similar gradients, and a filtering mechanism that chooses valuable gradients for aggregation. By adopting RFCaps, we observe a significant reduction in the impact of poisoning attacks, thereby enhancing the performance and convergence of the global model in federated learning environments. In the following section, we will explore the core components of the RFCaps algorithm in depth.

## 4.4 | Clustering Gradients Method of RFCaps

Worth noting that, in this paper, unlike other research in the field of accelerating the convergence of federated learning methods, we have assumed the existence of destructive attackers. While clustering-based methods are recognized for their efficacy in addressing bias introduced by non-IID data, they fall short when confronted with malicious attackers. In RFCaps, we delve into the examination of three prevalent attack types, revealing that gradient clustering techniques can suffer from reduced accuracy due to attackers directly manipulating weights. Such attacks undermine the presumed relationship between gradients and client data, demanding a more robust solution. In order to bolster clustering accuracy amid the presence of attackers, RFCaps strategically employs a smaller dataset containing just 10 samples per class. This enables us to meticulously evaluate gradients and execute the clustering process based on these assessment results. Notably, each model is assigned to the cluster that most accurately diagnoses its corresponding class during evaluation, achieved through the application of an argmax function. The initiation of this clustering transpires during the first communication round, capitalizing on purer gradients that faithfully mirror their intrinsic connection to the client's data.

## 4.5 | Optimal Weight Selection

In addition, to have a higher accuracy aggregation in the presence of attackers, we have used the loss obtained by each model at the clustering step as a measure of the usefulness of each model at the step of aggregation. This approach replaces random item selection with a dynamic process that selects the most valuable k weights from each cluster. The value of k adapts dynamically to maximize its effectiveness, enabling coordinating servers to efficiently leverage model weights and harness the potential of available device data.

One of our underlying assumptions is the potential escalation of attackers to nearly half of all available devices. Accordingly, we establish the value of the variable k by evaluating the average loss across each cluster. Initially, we arrange the gradients within each cluster in ascending order based on the loss incurred during the clustering step (each cluster is comprised of gradient-loss pairs). Subsequently, we compute the average loss of the latter half of the ordered list and exclusively opt for gradients with loss values lower than this computed average. Significantly, the variable K signifies the count of selected models upon implementation of this criterion. Consequently, this procedure serves as a filtering mechanism, effectively eliminating numerous attackers at this stage.

In the subsequent stage, we compute the mean loss of the gradients determined in the preceding step. Within each cluster, we proceed to choose the gradients whose loss values fall below this new average loss. These chosen gradients are then designated as the ultimate participants in the aggregation process, to be aggregated through the averaging method. Ultimately, the local models and global model weights will updated. This iterative process persists until convergence is achieved. Therefore, as we will demonstrate in the experiments section, the proposed method has a high resistance to the most severe types of attacks. This is because the attackers cannot affect the training process by altering their attack type or its severity. RFCaps aggregation algorithm is detailed in algorithm 1.

## 4.6 | Workflow

Figure1 illustrates how our proposed method, RFCaps, performs federated learning by choosing valuable gradients in each round, following the steps below:

**Step 1:** At the FL server, global model is created and initialized with random gradients

**Step 2:** All available devices connect to the FL server and download the global model weights $\omega_0$.

**Step 3:** Each device performs local training for one epoch and sends the resulting model weights to the FL server.

**Step 4:** On the FL server, in the first communication round, a model is created per each client's weights, and this model is evaluated by a minimal test set (10 samples per class). According to the evaluation, accurate clustering is conducted using the model's correct detection for each class. So, we have a dictionary whose keys demonstrate class numbers, and its values are a list of models gradients and their specifications, such as loss and accuracy. In other communication rounds, only specifications are updated.

**Step 5:** Per each cluster, the average loss of the second half of each cluster weights is calculated, finally per each cluster a list of weights is created whose loss is less than the cluster average loss. In addition, the average loss of all selected list is calcuted and the the average loss filter apply over each cluster again.

**Step 6:** In this phase, inner cluster aggregation per each cluster is conducted.

**Step 7:** Finally, in this step $\omega_{t+1}$ is calculated based on FedAvg. The steps 2-6 will need to be repeated until the desired accuracy is reached or until the desired number of communication rounds are reached.

**Algorithm 1** Robust Federated Learning Against Poisoning Attacks (RFCaps)

N ← The number of clients
T ← maximum number of communication rounds
C={c: [] for i in range(N)}
$K = 1$
$\omega_0 \leftarrow rand()\{Initialization\}$
**for** $t \leftarrow 1$ to $T$ **do**
Broadcast $\omega_t$ to all clients
Wait until all gradients $\{\omega_t^i : i \in N\}$ arrive
**for** $i \leftarrow 1$ to $N$ **do**
$(loss_i, ppc_i) \leftarrow$ eval($\omega_t^i$)
**end for**
**for** $i \leftarrow 1$ to N **do**
**if** $t = 1$ **then**
$CNO_i \leftarrow$ argmax($predict_i$ )
$C_{CNO_i} \leftarrow (w_t^i, loss_i, i)$
**else**
$updateLoss((w_t^i, loss_i, i))$
**end if**
**end for**

$$AVL_t \leftarrow \frac{\sum_j^{NC} \sum_l^K C_{j,l}[1]}{N}$$

$$G \leftarrow \sum_j^{NC} \sum_l^K C_{j,l}[0] < AVL_t : C_{j,l}$$

$$SAVL_t \leftarrow \frac{\sum_j^{len(G)} G_j[0]}{N}$$

$$FG \leftarrow \sum_j^{NC} \sum_l^K C_{j,l}[0] < SAVL_t : C_{j,l}[0]$$

$$\omega_{t+1} \leftarrow FedAvg(FG)$$

**end for**

**def eval** :
require gradients
TS ← a test set included 10 samples per each class.
(loss, correctPredictions) ← evaluate( model(gradients))
▹ This process involves constructing a model using the provided gradients and evaluating it with the TS minimal dataset. Subsequently, it enumerates the correct predictions for each class, creating a list of these enumerations, and computes the loss of the model. The outcome encompasses both the loss value and the enumerated list.

**def updateLoss** :
require ($\omega$, loss,index)
$\sum_i^{NC} \sum_j^{len(C_i)} c_{i,j}[2] == index : C_{i,j} = (\omega, loss, index)$
▹ This procedure iterates through all clusters, identifying the relevant element by its index, and subsequently modifying its specifications based on the provided arguments.

# 5 | EXPERIMENTS

In this section, we will review the performance of the proposed method. We have compared the proposed method with four widely known aggregation algorithms. and three types of well-known attacks that have been addressed in most articles in different intensities. It is important to highlight that the primary objective of this article is to offer a solution that guarantees utmost security and enhances convergence speed. Consequently, the aim of this paper is not solely to attain the highest accuracy compared to existing models. Put differently, depending on the specific application, alternative models can readily be incorporated into the proposed approach.

## 5.1 | Experimental Environment

The main objective of the experiments is to investigate the efficiency of RFCaps. We compare the performance of our approach against data poisoning attacks with the most prevalent methods in the federated learning literature, namely FedAvg [13], COMED [14], Trimmed Mean [14], and Multi-KRUM (MKRUM) [15] . All experiments were executed on an HP PC equipped with a 3.30GHz Intel Core i7-7020 processor, 12GB RAM, and a Tesla K80 GPU, running Ubuntu Desktop v18.04 LTS. we implement our algorithm in Python programming language and the Pytorch library version 1.8.

In this section, we evaluated the performance of our approach for RfCaps using two datasets. MNIST [32] dataset which has 70000 28 × 28 handwritten digits, and Fashion-MNIST[33] which is a dataset comprising 28×28 grayscale images of 70,000 fashion products. It is noteworthy to emphasize that these datasets have been conventionally employed to showcase the operational efficiency of established methodologies. As a result, they serve as essential reference points, against which we can rigorously assess the efficacy of our innovative approach. For the MNIST dataset, we employed a capsule neural network with a configuration of 16 kernels for the convolutional layer and 32 kernels for the PrimaryCaps layer. The kernels' dimensions were set at 9 x 9, while each capsule was represented as a vector with a length of 8. In the case of the Fashion-MNIST dataset, we adopted a similar approach but utilized 32 kernels for the convolutional layer and 64 kernels for the PrimaryCaps layer. Notably, the other architectural specifications remained consistent with those employed for the MNIST dataset.

To compare different methods, we focus on their accuracy achieved in each aggregation round. To handle the numerous comparisons, we created a summary chart. Each point on this chart shows the average accuracy of a method across 100 aggregation rounds. By averaging accuracy over these rounds, we get insights into the method's overall accuracy and stability. Indeed, aggregation methods that fail to achieve the desired accuracy during the intermediate stages of the training process tend to exhibit limited success in attaining a high average accuracy. In essence, these methods would not demonstrate superior average accuracy if they were susceptible to the influence of attackers, regardless of whether they attain peak accuracy in the concluding stages. Additionally, at the end of this section, we provide a detailed comparison and thoroughly assess how our method performs in comparison to other approaches.

## 5.2 | Evaluation Results

To assess the efficacy of the proposed method, we subjected it to evaluation using three distinct attack types: label-flipping attacks, Byzantine attacks, and noisy client attacks. Byzantine attacks are recognized as particularly potent since even the presence of a single attacker can significantly disrupt the entire training process [13]. The label-flipping attack, being relatively easier to execute, is a common choice for attackers with direct dataset access. Its impact can be varied by adjusting the extent of label-flipping, such as flipping multiple labels in each client's dataset. Moreover,

we introduced the noisy client attack to facilitate a more comprehensive assessment of the method's performance.

Furthermore, to assess the efficacy of the proposed method, we employed the FedAvg [13], COMED [14], Trimmed Mean [14], and Multi-KRUM (MKRUM) [15] algorithms as aggregation techniques for comparison and evaluation. Our evaluation also encompassed varying levels of attacker presence across five different intensity levels. This involved augmenting the potential for disruption by increasing the number of attackers. We conducted separate experiments with all algorithms for each attack type, incorporating 5, 10, 15, 20, and 24 attackers. Given the operational constraints of the MKRUM algorithm, which operates optimally with no more than half of the participants being attackers, we capped the maximum number of attackers at 24. The total number of clients remained at 50 throughout all experiments, and the non-IID rate of local datasets was maintained at 80%. These stringent experimental conditions were designed to rigorously gauge the robustness and stability of the algorithms under challenging training scenarios. In the subsequent subsections, we comprehensively evaluate each attack individually.

### 5.2.1 | Byzantine attacks

In this section, we will examine the impact of Byzantine attackers. The primary objective of this evaluation is to examine the effectiveness of our proposed method relative to other approaches in mitigating the effects of the Byzantine attack. The aim of this attack in this experiment is the reducing overall accuracy, so to intensify the detrimental impact of this attack, we have introduced a form of cooperative attack, wherein the attackers collaborate in generating the transmitted gradients. In this approach, the gradients from the attackers are averaged and then scaled by a constant factor ( $\lambda$ ), indicating the degree of deviation from the computed average. Through experimental analysis, we have determined this factor to be 0.16. Let "A" denote the set of Byzantine attackers' gradients, then the altered gradients ($\omega_a$) can be calculated using the following formula:

$$\omega_a = \frac{\sum_i^{len(A)} A[i]}{len(A)} * \lambda \tag{5}$$

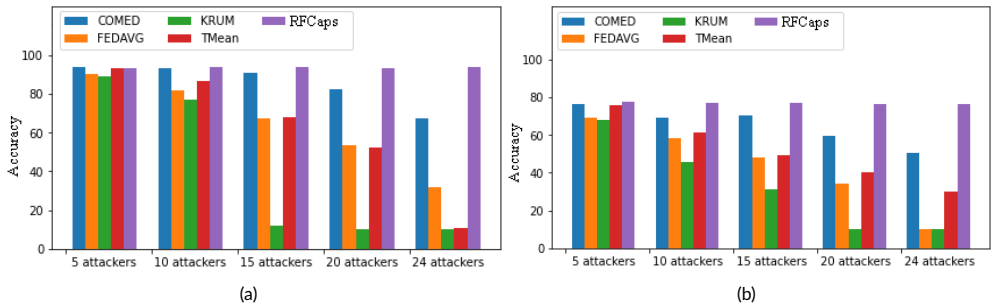


**FIGURE 2** Average Test Accuracy (%) across 100 rounds and 50 clients, with varying numbers of Byzantine attackers. Results shown for MNIST dataset in (a) and Fashion-MNIST dataset in (b).

Figure2 illustrates the outcomes of experiments conducted on the MNIST (a) and Fashion-MNIST (b) datasets involving 50 clients and 100 communication rounds. In this experiment, each group of bars corresponds to a different

level of attack intensity, and each chart consists of five intensity levels. Each bar in these charts represents the average accuracy (%) over 100 communication rounds.

### 5.2.2 | Label-Flipping attacks

In this section, we examine the impact of label-flipping attacks. The primary objective of this evaluation is to investigate the effectiveness of our proposed method relative to other approaches in countering the effects of label-flipping attacks. The aim of this attack in this experiment is the reducing overall accuracy. To this end, each attacker arbitrary changed the labels of two or three class of their own local datasets.In this experiment, both the criteria and the entire evaluation setup closely mirror those detailed in the Byzantine attack evaluation described in Section 5.2.1. Figure3 shows the results of experiments investigating label-flipping attacks.

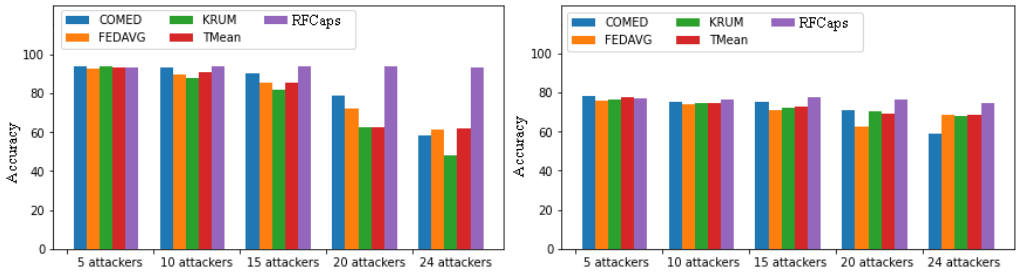


**FIGURE 3** Average Test Accuracy (%) across 100 rounds and 50 clients, with varying numbers of Label-Flipping attack. Results shown for MNIST dataset in (a) and Fashion-MNIST dataset in (b).

### 5.2.3 | Noisy Clients

In this section, we analyze the impact of noisy client attacks. Our primary objective is to examine the effectiveness of our proposed method in mitigating the effects of noisy client attacks, compared to other methods. The goal of this attack in our experiment is to lower the overall accuracy. To simulate this, we introduced uniform noise to all the pixels of the images in the attacker's dataset. This experiment shares the same criteria and evaluation setup as outlined in the Byzantine attack assessment in Section 5.2.1. Results were shown in Figure4.

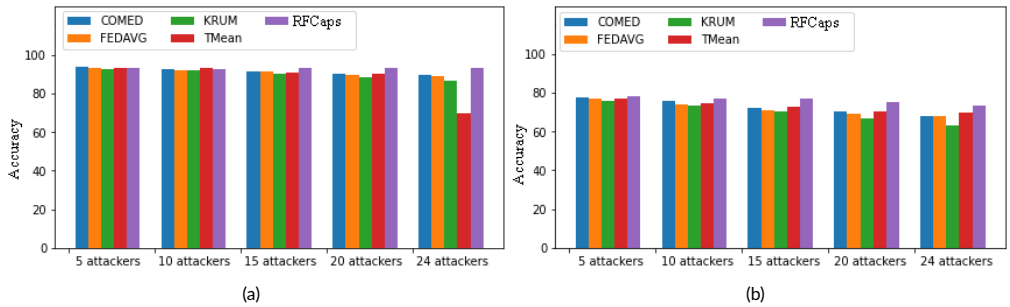


**FIGURE 4** Average Test Accuracy (%) across 100 rounds and 50 clients, with varying numbers of Noisy Clients attack. Results shown for MNIST dataset in (a) and Fashion-MNIST dataset in (b).

### 5.2.4 | Discussing the evaluation results

The robustness of our proposed method against various attacks, including label-flipping attacks, noisy client attacks, and Byzantine attacks, can be attributed to several key factors. Our approach utilizes a clustering-based strategy for the identification and grouping of gradients originating from diverse clients. This method involves evaluating the accuracy and performance of these gradients, contributing to the creation of clusters characterized by enhanced accuracy and reduced impact of non-IID data. This enhances the quality of the aggregated model, making it less susceptible to the noise introduced by the attacks. Furthermore, the selection of gradients for aggregation is based on their loss values. This strategic choice ensures that only the most valuable and reliable gradients contribute to the aggregation process, effectively filtering out the influence of gradients from attackers with high loss values. Additionally, we dynamically adjust the number of selected models for aggregation based on the ascending order of cluster losses. This adaptability ensures that only the gradients from reliable clients are used for aggregation, minimizing the impact of the attacks. In summary, our method's robustness against various attacks arises from the combination of accurate gradient clustering, loss-based gradient selection, and dynamic adjustment of aggregation participants. These elements collectively empower the method to effectively isolate and mitigate the influence of attacks, resulting in improved robustness and performance.

## 5.3 | Results in depth

In this section, we will illustrate the results of the previous experiments in more precise detail. In the Figure5 and Figure6, datasets, attacker types, and the number of attackers have been grouped in each chart. In other words, the rows belong to an attack type, and each column specifies the number of attackers. In addition, Table2 and Table3 show the accuracies obtained in different intensity of attacks. The results show that the proposed algorithm are superior to other algorithms in all experiments, or at least has the same accuracy with a bit of difference. In short, we will mention the notable differences. The results in Table2 demonstrate that the superior accuracies obtained by RFCaps with the presence of the most attackers are 80% with the KRUM algorithm, 71% with the TMEAN algorithm, 16% with the COMED algorithm, and 59% with the FEDAVG algorithm for Byzantine attacks; about 23% with COMED in the label-flipping attack and 19% with the COMED algorithm in the noisy attack in the MNIST dataset. These results for RFCaps based on the result in Table3 are 62% with the KRUM algorithm, 48% with the TMEAN algorithm, 20% with the COMED algorithm, and 72% with the FEDAVG algorithm for Byzantine attacks; about 15% with COMED in

the label-flipping attack and 8% for KRUM in the noisy attack in the Fashion-MNIST dataset.

Additionally, we assessed the effectiveness of RFCaps' clustering method in mitigating non-IID data bias through an experiment across varying non-IID data percentages. With 100 participants and a 10-sample-per-class test set for the clustering algorithm, the results highlighted the method's successful gradient clustering performance under standard non-IID data scenarios, supported by direct gradient assessment. The findings are showcased in Figure 7.

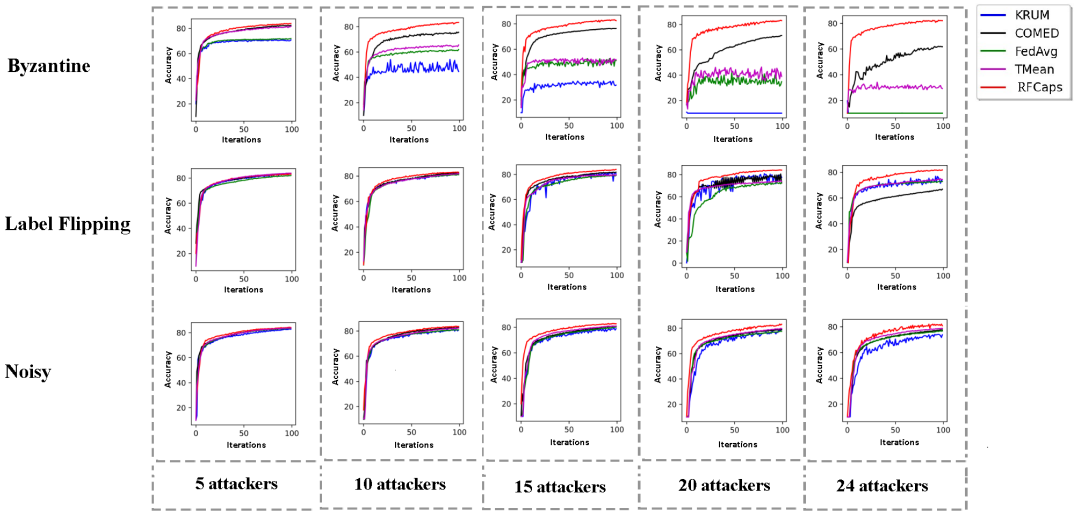


**FIGURE 5** In this experiment, which has been conducted using the Fashion-MNIST dataset each column shows the increasing severity of the attack based on the number of attackers, which are categorized into 5 groups including 5,10,15, 20, and 24 attackers respectively. In addition, each row illustrates the type of attacker: The Byzantine attack for the first row, the Noisy attack for the second row, and the Label-Flipping attack for the third row.

| | algorithms | 5 attackers | 10 attackers | 15 attackers | 20 attackers | 24 attackers |
|---|---|---|---|---|---|---|
| **Byzantine** | FEDAVG | 95.1 | 86.9 | 71.9 | 59.5 | 38.5 |
| | COMED | 97.9 | 97 | 95.5 | 92.1 | 81.1 |
| | TMEAN | 97.6 | 92.1 | 72.1 | 59.4 | 25.6 |
| | MKRUM | 94 | 80 | 24 | 21 | 17 |
| | RFCAPS | 97.6 | 97.7 | 97.7 | 97.5 | 97.6 |
| **Label-Flipping** | FEDAVG | 97.6 | 96.4 | 94.6 | 84.7 | 77.4 |
| | COMED | 97.8 | 97.4 | 96.5 | 93.8 | 80.7 |
| | TMEAN | 97.8 | 96.6 | 94.82 | 74.0 | 73.91 |
| | MKRUM | 97.7 | 96.8 | 97.0 | 96.0 | 91.0 |
| | RFCAPS | 97.7 | 97.8 | 97.7 | 97.6 | 97.5 |
| **Noisy** | FEDAVG | 97.6 | 97.3 | 97.2 | 96.75 | 96.1 |
| | COMED | 97.8 | 97.4 | 97.2 | 96.7 | 96.5 |
| | TMEAN | 97.5 | 97.2 | 97.0 | 96.8 | 78.6 |
| | MKRUM | 97.7 | 97.3 | 96.6 | 96.4 | 95.6 |
| | RFCAPS | 97.6 | 97.4 | 97.8 | 97.6 | 97.5 |

**TABLE 2** The highest accuracy achieved of each algorithm per 100 communication rounds on different levels of attack severity of MNIST dataset.

| | algorithms | 5 attackers | 10 attackers | 15 attackers | 20 attackers | 24 attackers |
|---|---|---|---|---|---|---|
| **Byzantine** | FEDAVG | 71.9 | 61.9 | 52.6 | 42.1 | 10.0 |
| | COMED | 82.4 | 75.7 | 76.3 | 71.3 | 62.1 |
| | TMEAN | 81.6 | 65.5 | 53.0 | 46.4 | 34.1 |
| | MKRUM | 71.2 | 54.1 | 34.9 | 10.8 | 19.6 |
| | RFCAPS | 83.9 | 83.4 | 83.2 | 83.4 | 82.4 |
| **Label-Flipping** | FEDAVG | 82.1 | 81.2 | 79.4 | 73.6 | 72.8 |
| | COMED | 83.8 | 82.6 | 81.8 | 80.5 | 66.8 |
| | TMEAN | 83.9 | 82.1 | 79.8 | 74.4 | 75.0 |
| | MKRUM | 83.4 | 81.8 | 81.7 | 80.8 | 76.8 |
| | RFCAPS | 83.3 | 83.1 | 84.0 | 84.2 | 81.7 |
| **Noisy** | FEDAVG | 83.4 | 81.1 | 80.1 | 77.7 | 76.8 |
| | COMED | 83.6 | 82.8 | 80.9 | 78.8 | 77.6 |
| | TMEAN | 83.7 | 82.3 | 81.1 | 79.8 | 78.7 |
| | MKRUM | 83.1 | 81.3 | 79.3 | 79.2 | 74.1 |
| | RFCAPS | 84.2 | 83.8 | 82.9 | 83.2 | 82.0 |

**TABLE 3** The highest accuracy achieved of each algorithm per 100 communication rounds on different levels of attack severity of Fashion-MNIST dataset.

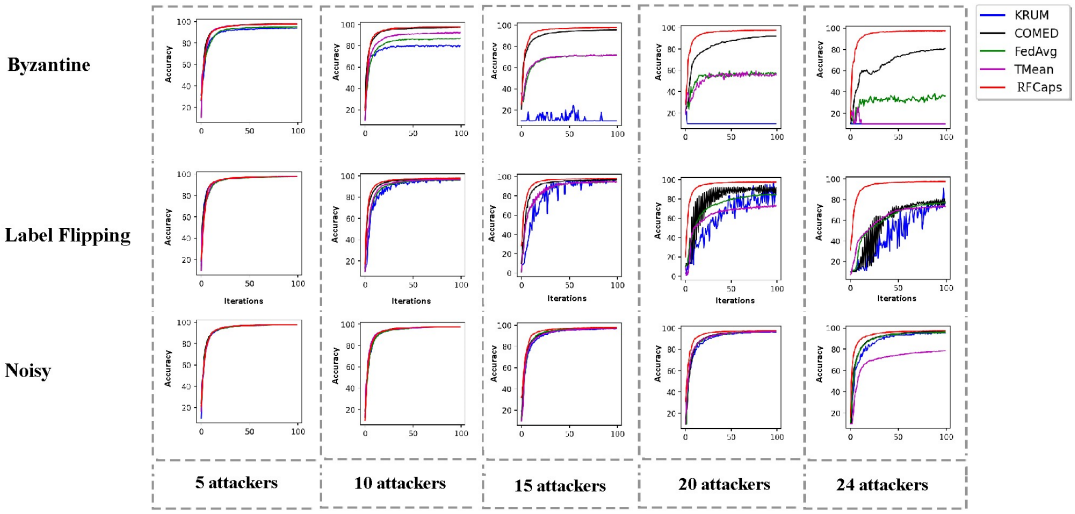


**FIGURE 6** In this experiment, which has been conducted using the MNIST dataset each column shows the increasing severity of the attack based on the number of attackers, which are categorized into 5 groups including 5,10,15, 20, and 24 attackers respectively. In addition, each row illustrates the type of attacker: The Byzantine attack for the first row, the Noisy attack for the second row, and the Label-Flipping attack for the third row.

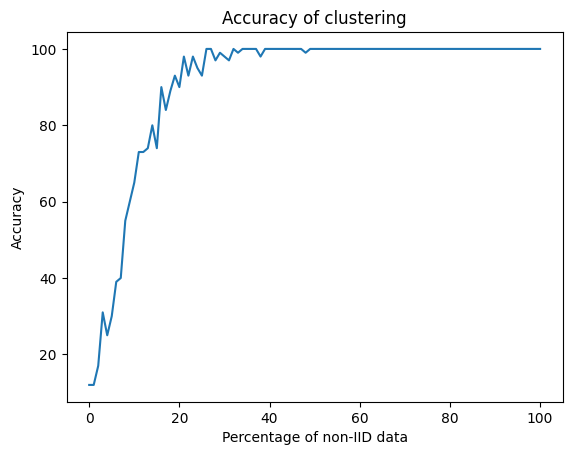


**FIGURE 7** Acuraccy of prediction-base clustering method used in RFCaps

## 6 | DISCUSSION

Reviewing prior research on the security aspects of the federated learning approach reveals a prevailing trend: many proposed aggregation algorithms possess transient efficacy, rendering them susceptible to attackers' evolving strategies. This underscores the need for a solution that offers a secure, rapid, and dependable federated learning framework. Our devised method, which intricately examines clients' gradients, showcases robust resistance against the most formidable threats posed by poisoning attacks. Notably robust and secure as it is, our approach does entail certain limitations, such as elevated time complexity due to the gradient examination process, which is relatively higher compared to other established algorithms. However, given the considerable computational prowess of coordinating servers, this concern becomes negligible, allowing us to harness the advantages brought by a vast number of clients. Furthermore, the contemporary formidable computational capacity of servers underscores the paramount significance of security in federated learning settings, reinforcing our ability to overlook the algorithm's inherent time complexity. Nonetheless, the time complexity of our algorithm can be effectively managed through the parallelization of the gradient examination process.

## 7 | CONCLUSION

By thoroughly investigating the challenges inherent in federated learning and conducting extensive experimentation to assess robustness and convergence dynamics, we have developed a resilient federated learning algorithm capable of countering poisoning attacks and handling non-IID data. This algorithm enhances the security and efficiency of federated learning, maintaining stability under high attack pressures and achieving convergence within fewer communication rounds. Furthermore, our proposed method was benchmarked against widely-used robust aggregation algorithms, consistently outperforming them in various experiments. Despite the evolving nature of attacks and their variations, our endeavor was to establish a secure foundation for federated learning. Through a swift and secure approach, we aimed to tackle the primary obstacles of federated learning, fostering its safe and widespread adoption. Ultimately, our proposed method aspires to allay apprehensions surrounding the implementation of federated learning, driving its acceptance and application in diverse scenarios.

## References

[1] Wang, H., Kaplan, Z., Niu, D., & Li, B. (2020, July). Optimizing federated learning on non- IID data with reinforcement learning. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications (pp. 1698-1707). IEEE.

[2] Jiang, J., & Hu, L. (2020). Decentralised federated learning with adaptive partial gradient aggregation. CAAI Transactions on Intelligence Technology, 5(3), 230-236.

[3] Muñoz-González, L., Co, K. T., & Lupu, E. C. (2019). Byzantine-robust federated machine learning through adaptive model averaging. arXiv preprint arXiv:1909.05125.

[4] Taheri, R., Javidan, R., Shojafar, M., Pooranian, Z., Miri, A., & Conti, M. (2020). On defending against label flipping attacks on malware detection systems. Neural Computing and Applications, 32(18), 14781-14800.

[5] Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. arXiv preprint arXiv:1710.09829.

[6] Li, H., Ota, K., & Dong, M. (2018). Learning IoT in edge: Deep learning for the Internet of Things with edge computing. IEEE network, 32(1), 96-101.96– 101, 2018.

[7] Mazzia, V., Salvetti, F., & Chiaberge, M. (2021). Efficient-CapsNet: Capsule Network with Self-Attention Routing. arXiv preprint arXiv:2101.12491.

[8] Lyu, L., Yu, H., & Yang, Q. (2020). Threats to federated learning: A survey. arXiv preprint arXiv:2003.02133.

[9] Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492.

[10] Dundar, M., Krishnapuram, B., Bi, J., & Rao, R. B. (2007, January). Learning classifiers when the training data is not IID. In IJCAI (Vol. 2007, pp. 756-61).

[11] Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., & Srivastava, G. (2021). A survey on security and privacy of federated learning. Future Generation Computer Systems, 115, 619-640.

[12] Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2), 1-19.

[13] Baruch, G., Baruch, M., & Goldberg, Y. (2019). A little is enough: Circumventing defenses for distributed learning. Advances in Neural Information Processing Systems, 32.

[14] Yin, D., Chen, Y., Kannan, R., & Bartlett, P. (2018, July). Byzantine-robust distributed learning: Towards optimal statistical rates. In International Conference on Machine Learning (pp. 5650-5659). PMLR.

[15] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., & Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. Advances in Neural Information Processing Systems, 30.

[16] Mazzia, V., Salvetti, F., & Chiaberge, M. (2021). Efficient-capsnet: Capsule network with self-attention routing. Scientific Reports, 11(1), 1-13.

[17] Chen, M., Mathews, R., Ouyang, T. and Beaufays, F., "Federated Learning of out-of-vocabulary words", arXiv preprint arXiv:1903.10635, (2019).

[18] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C. and Ramage, D.,"Federated learning for mobile keyboard prediction", arXiv preprint arXiv:1811.03604, (2018).

[19] Wang, Y., "Co-op: Cooperative machine learning from mobile devices", (2017).

[20] McMahan, B., Moore, E., Ramage, D., Hampson, S. and y Arcas, B.A., "Communication-efficient learning of deep networks from decentralized data", in Artificial Intelligence and Statistics, PMLR. 1273-1282.

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

[22] Xie, C., Koyejo, S. and Gupta, I., "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance", in International Conference on Machine Learning, PMLR., 6893- 6901.

[23] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D. and Shmatikov, V., "How to backdoor federated learning", in International Conference on Artificial Intelligence and Statistics, PMLR., 2938-2948.

[24] Sun, Z., Kairouz, P., Suresh, A.T. and McMahan, H.B., "Can you really backdoor federated learning?", arXiv Preprint arXiv:1911.07963, (2019).

[25] Ouyang, S., Dong, D., Xu, Y., & Xiao, L. (2021). Communication optimization strategies for distributed deep neural network training: A survey. Journal of Parallel and Distributed Computing, 149, 52-65.

[26] Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., & Goldstein, T. (2018). Poison frogs! targeted clean-label poisoning attacks on neural networks. Advances in neural information processing systems, 31.

[27] Gu, T., Dolan-Gavitt, B., & Garg, S. (2017). Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733.

[28] Biggio, B., Nelson, B., Laskov, P. (2012). Poisoning attacks against support vector machines. arXiv preprint arXiv:1206.6389.

[29] Fung, C., Yoon, C. J., & Beschastnikh, I. (2018). Mitigating sybils in federated learning poisoning. arXiv preprint arXiv:1808.04866.

[30] Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., & Tygar, J. D. (2011, October). Adversarial machine learning. In Proceedings of the 4th ACM workshop on Security and artificial intelligence (pp. 43-58).

[31] Zhu, H., Xu, J., Liu, S., & Jin, Y. (2021). Federated learning on non-IID data: A survey. Neurocomputing, 465, 371-390.

[32] Y. LeCun, "The MNIST database of handwritten digits," http://yann. lecun. com/exdb/mnist/, 1998.

[33] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," arXiv preprint arXiv:1708.07747, 2017.).