RAFL: A Robust and Adaptive Federated Meta-Learning Framework Against Adversaries

Zhengxin Yu, Yang Lu, Neeraj Suri
School of Computing and Communications, Lancaster University, United Kingdom
Email: {z.yu8, y.lu44, neeraj.suri} @lancaster.ac.uk

***Abstract*—With the emergence of data silos and increasing privacy awareness, traditional centralized machine learning provides limited support. Federated learning (FL), as a promising alternative machine learning approach, is capable of leveraging distributed personalized datasets from multiple clients to train a shared global model in a privacy-preserving manner. However, FL systems are vulnerable to attacker-controlled adversarial clients that potentially conduct adversarial attacks by uploading unreliable model updates or clients unintentionally uploading low-quality models leading to degraded FL performance and reduced resilience to attacks. In this paper, we propose RAFL: a new robust-by-design federated meta learning framework capable of mitigating adversarial model updates on non-IID data. RAFL leverages 1) a residual rule-based detection method and a Variational AutoEncoder (VAE) learning based detection method combined to distinguish adversarial clients from benign clients. 2) a similarity-based model aggregation method to reduce the likelihood of uploading adversarial models from adversarial clients. 3) multiple learning loops to collaboratively train multiple personalized detection models against adversaries effectively. Experimental results demonstrate that our proposed FL framework is robust by design and outperforms other defensive methods against adversaries in terms of model accuracy and efficiency.**

***Index Terms*—Federated Learning, Deep Learning, Adversarial Client Detection, Robust Aggregation**

I. Introduction

Federated Learning (FL) is a Machine Learning (ML) approach whereby multiple clients (*e.g.,* mobile devices, autonomous cars) collaboratively train a global ML model without sharing individual client data [1]. FL has been adopted across applications such as finance [2], healthcare [3] and Multi-access Edge Computing (MEC) [4]. In a typical FL scenario, a central server periodically updates a global model via collecting and aggregating local client models. As no client data is exchanged during the whole process, FL can inherently provide privacy and security guarantees to participating clients.

However, the quality of global models trained by FL may be degraded if clients upload inaccurate local models. Such inaccurate uploads can be due to non-deliberate failures from clients and/or adversarial attacks by corrupted clients [5]. Non-deliberate failures may be caused by noisy training labels, poor training datasets, or unintentional upload of low-quality local models from high mobility [6] [7]. Alternately, corrupted clients could distort their local model uploads by intentionally launching various data poisoning attacks, *e.g.*, Byzantine attacks [8] and backdoor attacks [5] [9].

Two prominent defensive approaches are typically advocated to deal with adversarial attacks in FL, namely, robust learning and adversarial client detection. The robust learning-based approaches try to ensure a lower bound on the global model learning accuracy whilst tolerating a subset of corrupted clients [8] [10] [11] [12] [13]. In contrast, the adversarial client detection-based approaches first identify potential corrupted clients by examining abnormal local model uploads, and then remove the local models of these clients over the global model aggregation [14] [15] [16].

While the above two approaches have been shown to be effective in enhancing FL security for a variety of applications, both have specific limitations. Typical robust learning methods require FL setting assumptions such as a limited number/type of adversarial clients [8] [10] [12], or prior knowledge of client data distributions (*i.e.*, independent and identical distribution (IID)) [11] [13]. However, such assumptions are infeasible in dynamic scenarios. These adversarial client detection methods require access to sanitized public datasets for detector mechanism training [14] [16] or static rules to predict client model updates [15]. However, using public datasets contrasts to FL's goal of protecting privacy of individual clients' data. Moreover, FL clients commonly exhibit substantial join/leave behavior during the training process, thus applying static rules-based methods is infeasible. Such methods are unable to detect adversarial clients exhibiting moving target behavior via learning predictor rules or injecting late-time faults within the FL communication round to avoid detection. Such approaches result in degraded model accuracy and reduced resilience to attacks, and incur communication overhead between the server and clients.

In this paper, we propose a new Robust and Adaptive Federated meta Learning framework (RAFL) capable of reducing adversarial and unintentional low-quality model updates on non-IID data without any prior dataset knowledge. The framework comprises three complementary techniques to enhance FL robustness by combing adversarial client detection and robust learning. For detection, we propose a residual rule-based method to distinguish adversarial clients from benign clients by leveraging key characteristics in the rank domain of model updates. The detected adversarial clients are removed and remaining clients are clustered based on model updates for different applications. An online variational autoencoder (VAE) based detection model is proposed to further detect and remove the imperceptible abnormal model updates based on their low-dimensional embeddings in latent space. This is trained using collected sanitized dataset from rule-based methods. In latent space, the reconstruction error of adversarial clients are much larger than typical clients, thus adversarial clients can be easily differentiated. Using VAE achieves finer-grained detection via hidden features from model updates.

These two detection methods are integrated into a framework with a robust *learning* and a robust *model aggregation* components. We combine meta-learning with FL elements to provide a robust-by-design framework by generating multiple learning loops with each loop focusing on discrete data clusters to train bespoke learning-based detection models. To further enhance robustness, a similarity-based model aggregation algorithm is developed to aggregate a meta-model in each cluster by using Canonical Correlation Analysis (CCA) similarity to decide the aggregation weight of participating clients. To summarize, our main contributions are as follows:

- A robust-by-design federated meta-learning architecture is proposed to adaptively defend against a range of adversarial attacks.
- A composite rule-based and learning-based detection method to effectively identify adversarial clients via ranking domain and low-dimensional embeddings.
- An adaptive model aggregation method is proposed to aggregate the global model by considering the degree of similarity between the meta-model and calculated mean model to resilience attacks.

The rest of this paper is organized as follows. Section II reviews the related work. The system and threat models of RAFL are presented in Section III. Design details of the RAFL are described in Section IV. Evaluation results are given in Section V. Finally, Section VI concludes this paper.

II. Related Works

Robust Federated Learning: Robust FL-based works leverage selected statistics of local model updates to introduce additional rules in the global model aggregation/update to facilitate model accuracy in the presence of attackers. Blanchard *et al.* [8] proposed an aggregation rule (termed Krum) to select a simple model update among clients' model updates to aggregate the global model by using Euclidean distances. Xie *et al.* [10] developed a median-based robust aggregation rule (Medoid) for synchronous Stochastic Gradient Descent (SGD) with low time complexity and provable convergence to critical points. Yin *et al.* [11] designed a coordinate-wise aggregation rule (Trimmed Mean) to aggregate the model update of each coordinate separately. Chen *et al.* [12] designed a variant of the classical gradient descent method based on the geometric median of means of the gradients (GeoMed). Li *et al.* [17] introduced an additional $l1$-norm regularization on the cost function to achieve robustness against Byzantine attacks in distributed learning. Such approaches may result in a biased global model or limited mitigation against adversarial attacks.

Detection-based Federated Learning: This category of research has developed detection-based methods which can distinguish adversarial from benign clients and remove adversarial clients in model aggregation. Shen *et al.* [18] proposed a detection-based approach for backdoor attacks in collaborative ML. Sun *et al.* [9] designed a low-complexity defense mechanism that mitigates the impact of backdoor attacks in FL tasks through model weight clipping and noise injection. Li *et al.* [14] presented a variational autoencoder model based detector to detect and remove adversarial. As an extension of [14], Gu *et al.* [16] utilized a conditional variational autoencoder to distinguish adversarial clients from benign clients. Most of these methods require a clean public dataset in the server to train a basic model, which is in contrast to the original goal of FL. Zhang *et al.* [15] relied on sizeable historical data on client model updates to perform prediction leveraging static rules. However, in real-world scenarios, clients are constantly changing, which makes it hard to predict their model updates.

III. System and Threat Model

System model: Our goal is to identify adversarial clients and eliminate their negative impact on FL model training performance.We consider a conventional FL system model, comprising a MEC server (*e.g.*, a master autonomous car, a base station, or roadside unit) and N clients (*e.g.*, autonomous cars, mobile devices or unmanned aerial vehicles). Clients generate their own local datasets, $\mathcal{D}_1, ..., \mathcal{D}_N$. At each communication round, the MEC server randomly selects a subset of clients to train a shared global model on their respective local datasets. Due to data privacy concerns, clients will only upload their model updates to the MEC server.

Threat model: FL is vulnerable to adversarial attacks due to adversarial clients uploading unreliable model updates. The objective of adversarial clients is to send low quality model updates aiming to reduce the model performance of the final shared global model. We present the threat model adopted in this paper by considering the capabilities of adversarial clients as: (i) Any client can be adversarial; (ii) Adversarial clients are capable of continuously participating within the FL training process identical to a benign client [19]; (iii) Adversarial client can collude with other adversarial clients within current and future communication rounds [6]; and (iv) Adversarial clients can access participating client model architecture, parameters and dataset knowledge (size, type, distribution) within any FL communication round [6] [14] [15].

The constraints on the threat model are: (i)The upper bound on the number of adversarial clients is not more than 50% of the total clients [14] [16] [20]; ii) Adversarial clients can only know a stale version of the model [6]; (iii) The server is assumed to be benign [8] [21]. The threat model captures diverse adversarial behaviors and covers multiple attack types, including Sign Flipping [22], Additive Gaussian Noise [14], A Little is Enough Attack [23], and Zero Gradient [20].

IV. Robust and Adaptive Federated Meta Learning Framework (RAFL)

Following an overview of the RAFL approach, we present our rule-based and learning-based detection methods to identify and remove adversarial clients from the framework. We describe the similarity-based model aggregation to further reduce the negative impact of adversarial clients. Our framework

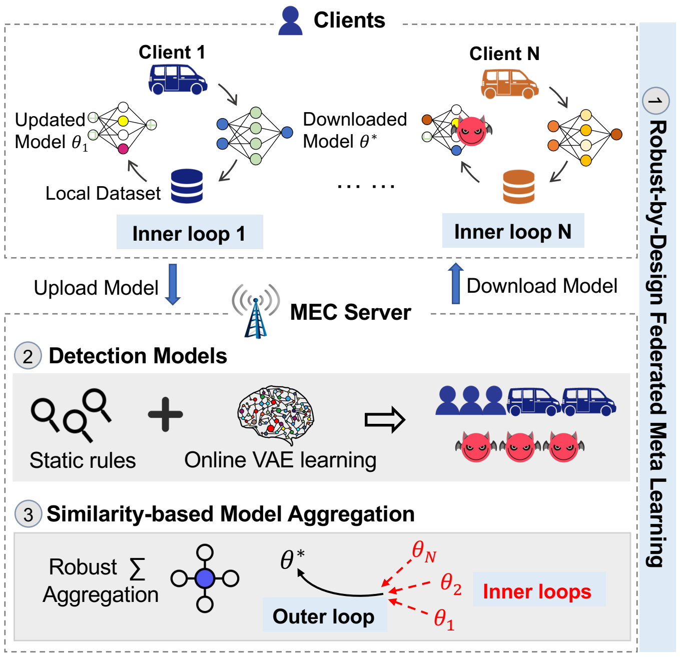

Fig. 1. RAFL System Architecture

comprises three components: **Federated Meta Learning, Detection, and Model Aggregation**, as shown in Fig. 1.

1) **Federated Meta Learning (Sec IV-A)**: Clients collect local data to perform model training to collaboratively train a shared global model in the MEC server by utilizing inner and outer loops. However, FL is susceptible to adversarial attacks due to adversarial clients uploading unreliable model updates resulting in degraded model performance.

2) **Detection (Sec IV-B)**: To defend against adversarial attacks, we combine a rule-based and learning-based adversarial detection model to identify adversarial clients in the MEC server. The rule-based detection leverages a residual-based detection model via a ranking matrix to rapidly detect adversarial client behavior. The learning-based detection using VAE online learning model to further identify abnormal model updates according to their low-dimensional embeddings.

3) **Model Aggregation (Sec IV-C)**: In the outer loop, a similarity-based model aggregation method is designed to conduct a global meta-model to further reduce the likelihood of uploading adversarial models from clients.

These three components combine robust learning with adversarial client detection to enhance FL robustness. Federated meta learning and model aggregation provide for robust learning and a framework-level defense against adversarial clients. For adversarial client detection, the rule-based and learning-based detection methods use different approaches to detect adversarial clients, consequently providing different detection granularity. The rule-based method detects adversarial clients by analyzing clients' parameter dimensions, while an online VAE detector utilizes extracted client features to identify adversarial clients. The VAE model is trained online based on the dataset collected from the rule-based method.

A. Federated Meta Learning

RAFL leverages local computation resources from clients and the MEC server/leader client for training personalized detection models on non-IID data against adversaries and rapidly adapting to new environments. The framework consists of associating discrete training loops to each of the decentralized data clusters. This is achieved via having separate learning loops for model training: Inner loop for task-level and Outer loop for cluster-level [24], as outlined in Algorithm 1. The **Inner loop** trains the task-specific detection model and executes it on the client device, from line 8 to 11. The **Outer loop** trains a cluster-level meta detection model which extracts common features from all task-level detection models within each cluster and executes it on the MEC server, from line 2 to 6 and 12 to 33. In our scenario, the clients are grouped into k disjoint clusters based on different applications since clients may use different applications/services, resulting in heterogeneous data distributions. The learning tasks in each client are correspondingly allocated into K clusters. The learning tasks $\mathcal{T}$ in K clusters are defined as $\widehat{\mathcal{T}} = \{\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^K\}$. Among all clusters, the multiple personalized detection models are trained by dynamically generating different learning loops (Outer loop and Inner loop) as well as addressing non-IID data. The outer loops extract common features from inner loops, while inner loops train task-specific models in clients.

Consequently, RAFL provides a robust-by-design framework given that these discrete inner/outer learning loops execute model training independently. Consequently, RAFL can provide sustained FL performance even if a few learning loops fail. For example, if some inner loops cannot work properly, other inner loops in the cluster can upload their model updates to the MEC server to aggregate a new global model and the MEC server can send the new global model to failed learning loops at the next communication round in order to start the new model training. If some outer loops fail, other outer loops at different MEC servers can share their meta-model with the failed outer loops.

Inner loop. A task-level model training, each client has a learning task $\mathcal{T}_i^k$ with its local dataset $\mathcal{D}_i$. Thus, a training model $f(\theta_i, \mathcal{D}_i)$ parameterized by θ is as follows [24]:

$$\theta_i^* = \underset{\theta_i \in R}{\arg\min}\, \mathcal{L}_{in}(\mathcal{D}_i; \theta_i, w), \tag{1}$$

where $\mathcal{L}_{in}$ is an inner loop (task-level) loss function to measure the error between true labels and predicted labels. w is the parameters of meta model in the outer loop which specifies assumptions about how to learn. θ is a p-dimensional parameter vector. θ^* is the updated local model. The target function of inner loop is to minimize $\mathcal{L}_{in}$. We define the loss function $\mathcal{L}_{in}$ as in [25]:

$$\mathcal{L}_{in} = f_i(\theta_i) + \frac{\lambda}{2}\|\theta_i - w\|^2, \tag{2}$$

where λ controls the contribution of w to the task-specific model. It is a regularization parameter by utilizing a regularized loss function with l_2-norm for each client. This loss function applied the first-order approximation method

Algorithm 1: RAFL runs on N clients (indexed by i).

```
Federated Meta Learning:
Outer loop:
$N$ clients are grouped into $K$ clusters
for each cluster $k \in \{1, \cdots, K\}$ in parallel do
    MEC server send $w$ to all clients
    $w_k = w_{k,0}$
    for each communication round $t \in \{1, \cdots, T\}$ do
        Inner loop:
        for client $i \in \{1, \cdots, N\}$ in parallel do
            $\theta_i^t \leftarrow \theta_i^{t-1} - \alpha \nabla \mathcal{L}_{in}\left(\theta_i^{t-1}\right)$
        end
        Adversarial Client Detection:
        if $t\%10 == 0$ then
            Convert all $\theta_i^t$ to $R$ matrix
            Compute $e$ and $v$ of the column in $R$
            Run $K$-means with $k = 2$
            Remove adversarial clients $i$
            Send $\theta_i^t$ to MEC server
            Save sanitized dataset $D_{VAE}$
        else
            Utilize $D_{VAE}$ to train VAE model
            VAE are updated by SGD based on Eq. (10)
            $\boldsymbol{z}_i = \mu + \sigma \odot \epsilon$
            Remove adversarial clients
            Send client model update $\theta_i^t$ to MEC server
        end
    end
    Similarity-based model aggregation:
    $w_k^t \leftarrow (1-\beta) w^{t-1} + \beta \sum_{i=1}^{N} \frac{\theta_i^{t-1}}{|\mathcal{D}_i|} * \varepsilon.$
    Update the global model in MEC server
    $w_{k,0} = w_k^t$
end
Return $w_{k,0}$
```

to remove the second derivative when backpropagating the meta-gradients, which may cause large communication and computation costs.

To obtain θ_i^*, RAFL optimizes this loss through SGD [26]:

$$\begin{aligned} \theta_i^t &= \theta_i^{t-1} - \alpha \nabla \mathcal{L}_{in}\left(\theta_i^{t-1}\right), \\ \text{s.t. } \theta_i^0 &= w, t = 1, 2, 3, \ldots T, \end{aligned} \tag{3}$$

where t is the current communication round in the FL model training. T is the total number of communication rounds. The initialization of the model θ_i^1 in the inner loop for the first communication round is the parameters of meta-model w in the outer loop.

Outer loop. For Cluster-level model training, the aim of the outer loop is to learn a general purpose meta-model w that can generalize across tasks within each cluster. The trained meta-model enables new tasks to be learned faster and more effectively than typical ML algorithms. Most importantly, meta-models can support each task to achieve a personalized model. The loss function of outer loop can be expressed as:

$$\begin{gathered} \mathcal{L}_{ou} = \frac{1}{N} \sum_{i=1}^{N} F_i(w), \\ \text{where } F_i(w) = \min_{\theta_i \in R} \left\{ f_i(\theta_i) + \frac{\lambda}{2} \left\| \theta_i - w \right\|^2 \right\}. \end{gathered} \tag{4}$$

The outer loop to update the meta-model is as:

$$w^t = (1-\beta) w^{t-1} + \beta \sum_{i=1}^{N} \frac{\theta_i^{t-1}}{|\mathcal{D}_i|}, \tag{5}$$

where β is a parameter to decide the weight of cluster-level model updates from the last communication round. The data size of each client has also been considered when calculating the updated model updates in the new communication round.

B. Adversarial Clients Detection

To defend against adversarial attacks, we combine a rule-based and learning-based adversarial detection model to identify adversarial clients. Static rule-based detection can be applied to detect adversarial clients without prior training, however exhibits weak adaptability to changing scenarios and cannot detect abnormal client behavior such as moving targets. Learning-based methods can adapt to dynamic scenarios yet are generally data-intensive (need large datasets) and require longer training time. As such, we have combined both approaches to address to overcome the aforementioned weaknesses when performing detection. Rule-based detection is deployed within the MEC server and leverages a residual-based detection model via a ranking domain to roughly detect adversarial client behavior and collect training data for learning-based method. The learning-based detection uses a VAE model to further identify abnormal model updates according to their low-dimensional embeddings, and resides within clients or MEC server. Each cluster trains a VAE detector online. We introduce the training process, respectively, of a rule-based model and VAE-based online detection model.

Residual Rule-based Detection Model: All participating clients sent their model updates to the MEC server to aggregate the global model, but these model updates may contain adversarial model updates. Inspired by [20], we utilize a ranking matrix R to detect adversarial model updates, since adversarial model updates have different key characteristics of the rank domain (*e.g.*, abnormal mean and standard deviation). As shown in Fig. 2, we first combine all model updates to a model matrix, $M \in \mathbb{R}^{n \times p}$. We assume that there are N participating clients and each client train a model parameterized by θ which is a p-dimensional parameter vector. The model matrix M can be defined as:

$$M_{i,:} = \frac{\left\| \theta_i^* - \theta_i^0 \right\|}{N_i}, \tag{6}$$

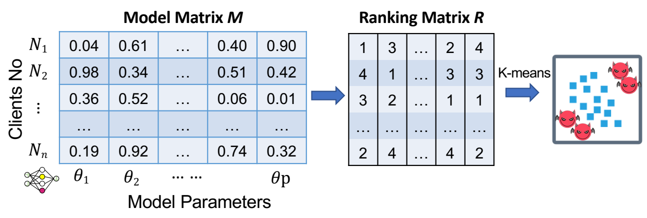

Fig. 2. Rule-based detection

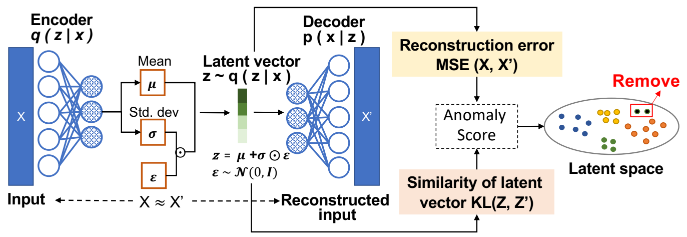

Fig. 3. Variational AutoEnocder Model

which is determined from

$$\begin{aligned}\theta_i^* &= \theta_i^0 - \frac{1}{N_i}\sum_{i=1}^{N_i}\frac{\partial f\left(\theta_i, \mathcal{D}_i\right)}{\partial\theta}, \\ \theta_i^0 - \theta_i^* &= \frac{1}{N_i}\sum_{i=1}^{N_i}\frac{\partial f\left(\theta_i, \mathcal{D}_i\right)}{\partial\theta},\end{aligned} \tag{7}$$

where the locally trained model is parameterized by θ, θ^* is the updated local model. We then transfer the M matrix to a ranking matrix R. Each column in M maps its elements to its respective permutation, ranked in descending order *e.g.*, $R(2.3, -3.1, 1.8) = (1, 3, 2)$. $M_{:,j}$ replaces the corresponding $R_{:,j} = R(M_{:,j})$ to retain its original vector position [20].

Next, we utilized the ranking matrix R to define the mean and variance of $R_{i,:}$ as follows:

$$e_i = \frac{1}{p}\sum_{j=1}^{p} R_{i,j} \quad \textit{and} \quad v_i = \frac{1}{p}\sum_{j=1}^{p}\left(R_{i,j} - e_i\right)^2. \tag{8}$$

Based on the calculated e and v, K-means is applied to cluster clients into two classes (benign clients and adversarial clients). We assume benign clients constitute the majority. In this way, adversarial clients can be easily found, however this detection model follows static rules which cannot detect moving target adversarial or imperceptible clients. Thus, we first remove the detected adversarial clients, and then the rest client model updates are collected to generate a relatively sanitized public model updates dataset D_{VAE} to train an online learning model to further detect imperceptible adversarial clients. The pseudocode of the rule-based detection is presented in the Algorithm 1 lines 14-19.

Variational AutoEnocder based Online Detection Model: We propose an online learning-based Variational AutoEncoder (VAE) detector to identify adversarial model updates based on reconstruction error and similarity of latent vector in latent space. Adversarial clients can be easily detected through VAE model, since the essential features of adversarial clients in latent space are substantially different from benign clients. In latent space, only essential features are retained and irrelevant features are removed [14]. VAE is an unsupervised learning model that extracts features from low-dimensional embeddings by copying its inputs to outputs. The VAE architecture as shown in Fig.3 consists of an encoder and a decoder. The encoder takes x as input and its output is a hidden representation z that is the low-dimensional embeddings. The decoder then utilizes these embeddings to reconstruct x and generate a reconstruction error used to optimize the parameters of the encoder-decoder model until it converges.

In VAE, the low-dimensional embeddings retain key features captured from uploaded clients' model updates. Moreover, VAE can remove noisy and redundant features within low-dimensional latent space as well as easily differentiate adversarial client embeddings due to their large reconstruction errors. We use the obtained sanitized public model updates dataset D_{VAE} from the rule-based detection model for VAE model training. The VAE model can recognize adversarial model updates as they trigger much higher reconstruction errors than benign model updates. We train the VAE model online over a newly updated dataset D_{VAE} every few communication rounds since clients constantly join/leave FL training. In this way, our detection model can quickly adapt to new environments and detect new attack variants. In our experiments, we execute the rule-based model every 10 communication rounds to obtain the new dataset D_{VAE}.

Specifically, we denote the encoder as $q_\psi(z \mid x)$, where ψ is weights and biases. The lower-dimensional space is stochastic. The output of the encoder is a Gaussian probability density. The representations z are sampled from this distribution and then feed z to the decoder to generate new samples that follow the original data distribution. The decoder is denoted by $p_\phi(x \mid z)$, with its input as z representation, and outputs parameters to data probability distribution of X, and weights and biases ϕ. The objective of the VAE training process is to maximize the probability of each x, given by:

$$p(x) = \int p(x \mid z)\, p(z)\, dz \tag{9}$$

The VAE loss function is the negative log-likelihood with a regularizer. The loss function l_i for datapoint x_i is:

$$\begin{aligned} l_i(\psi, \phi; x_i) = & -\mathbb{E}_{z\sim q_\psi(z|x_i)}\left[\log p_\phi(x_i \mid z)\right] + \\ & \mathbb{KL}\left(q_\psi(z \mid x_i) \parallel p(z)\right).\ell(\psi, \phi; x_i) \\ = & -\mathbb{E}_{z\sim q_\phi(z|x_i)}\left[\log p_\psi(x_i \mid z)\right] + \\ & \mathbb{KL}\left(q_\phi(z \mid x_i) \parallel p(z)\right). \end{aligned} \tag{10}$$

In order to train a VAE, two losses are used: reconstruction loss and similarity loss. These two losses also decide the threshold for adversarial client detection. The first term of Eq. 10 is the reconstruction loss, used to measure how much information is lost. It aims to learn a reconstruction as lossless as possible. In other words, reconstruction loss is the mean squared error (MSE) loss of the input and reconstructed output. The second term of Eq. 10 is similarity loss, which is the Kullback-Leibler (KL) divergence between the distribution of encoder $q_\psi(z \mid x)$ and $p(z)$. This divergence can measure how close q is to p. It forces the latent space's distribution to be similar to the assumed standard Gaussian distribution (zero mean and unit variance), $p(z) = N(0, 1)$.

The latent vector z is drawn from the encoder generated distribution, which is a random sampling. It makes difficult for backpropagation, since we cannot trace back errors in the encoder. To solve this problem, we use reparameterization to model the sampling process allow errors to propagate through the encoder network. Latent vector z can be rewritten as $z \sim p_\psi(z|x) = \mathcal{N}\left(\mu, \sigma^2\right)$ into $z = \mu + \sigma \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, 1)$ [27], where μ is the mean, σ is the variance of Gaussian distribution, and ϵ is an auxiliary noise variable.

Stochastic gradient descent (SGD) is used to optimize VAE with regard to the parameters of encoder ψ and decoder ϕ. The encoder parameters are updated using $\psi \leftarrow \psi - \rho \frac{\partial l}{\partial \psi}$. Similarly, the decoder is $\phi \leftarrow \phi - \rho \frac{\partial l}{\partial \phi}$ with step size ρ of SGD. The pseudocode of the VAE-based detection is shown in the Algorithm 1 lines 21-25.

Dynamic Threshold for Adversarial Client Detection: We applied the VAE-based detection model in every FL communication round to detect adversarial updates after executing the rule-based detection model. Due to the dynamic environment, we set a dynamic detection threshold to detect adversarial updates by considering both reconstruction error and similarity of latent vector (Anomaly score = reconstruction error + similarity loss), as we discussed in VAE detection model. It can dynamically calculate the average value of all reconstruction errors and similarity loss. Compared with this dynamic threshold, clients with larger reconstruction errors and similarity losses are recognized as adversarial clients and excluded from the aggregation step within the outer loop. Only benign clients participate in the aggregation process to conduct the global meta-model. The details of the aggregation method are described in Sec. IV-C and Algorithm 1 lines 28-29.

C. *Similarity-based Robust Model Aggregation.*

To further reduce the likelihood of uploading low-quality model updates from clients, we propose a Canonical Correlation Analysis (CCA) similarity-based model aggregation algorithm, as shown in Fig.4. It applies to the model aggregation step in the outer loop during the FL training process. Firstly, we calculate the mean value of model updates θ_{avg}. Then, we utilize CCA to achieve similarity scores. CCA can measure representation similarities between the calculated mean model updates and each client model updates by calculating distances between DL models [28]. This calculated distance is used as the similarity score. The similarity score decides the weight of the client when executing the model aggregation. The larger weight is given to the higher similarity score. We use ε to represent the similarity score which is converted to a percentage. Finally, the model aggregation in each outer loop

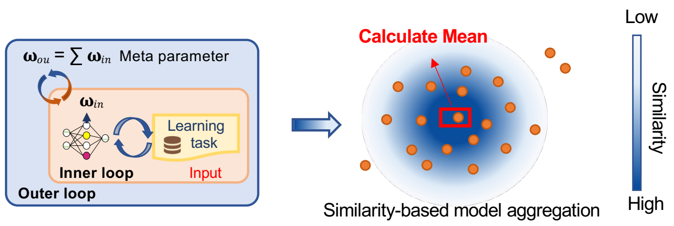

Fig. 4. Similarity-based model aggregation

can be expressed as:

$$w^t = (1-\beta)\, w^{t-1} + \beta \sum_{i=1}^{N} \frac{\theta_i^{t-1}}{|\mathcal{D}_i|} * \varepsilon, \qquad (11)$$

as shown in Algorithm 1 lines 29-31.

All of these aforementioned methods operate in tandem enabling the RAFL framework to operate. The complexity of both rule-based and VAE-based detection models is $O(q)$. The time complexity of RAFL is roughly linear to the number of global meta-model parameters.

V. Performance Evaluation

This section presents the experimental results of RAFL. We evaluate the performance of RAFL in image classification tasks with three models over three public datasets under four attacks. We demonstrate the effectiveness of our method by comparing it with five reference methods.

A. *Experiment Settings*

To evaluate the performance of our proposed RAFL framework, we consider an edge computing environment with multiple MEC servers coordinating 100 clients, executing within a HP Z440 workstation (i9 processor and GTX 3090). The computational capabilities of clients are heterogeneous. All 100 clients contain local datasets which are used to perform model training. The model aggregation is conducted on one of the MEC servers. Our experiments are performed on ML frameworks PyTorch v1.11.0 with CUDA v11.3.

B. *Datasets*

We utilize three datasets to evaluate our proposed RAFL for image classification tasks. These datasets have been widely used as benchmarks in FL [12] [14] [15].

MNIST: A 0-9 handwritten digits dataset, containing a training set of 60,000 examples and a test set of 10,000 examples. We shuffle data into 200 shards and each shard consists of 300 data examples. Each client is randomly assigned with 2 shards to simulate a heterogeneous environment setting with Non-IID data distribution.

Cifar-10: A 80 million color image dataset with labels, consisting of 60,000 32×32 images in 10 classes. Similar to the setting of MNIST dataset, we assign 2 shards to each client for a heterogeneous FL setting.

C. ML models

Logistic Regression (LR): A supervised learning classification algorithm used to predict the probability of a target variable. We use the MNIST dataset to train an LR model.

Deep Neural Networks (DNN): A neural network with multiple hidden layers between the input and output layers. In our experiments, the model architecture is $784{\times}100{\times}10$, trained by the MNIST dataset.

Convolutional Neural Network (CNN): A class of neural networks for processing data that has a grid-like topology. It consists of convolutional layers, pooling layers, and fully connected layers. The model in our experiments is trained by using Cifar10 with 2 CNN layers ($5{\times}5{\times}32$ and $5{\times}5{\times}64$), followed by a dense layer with 2048 units.

Variational AutoEncoder (VAE): VAE is our learning-based anomaly detector in RAFL, as discussed in Sec. IV. It is a generative model based on latent variables, consisting of an encoder and a decoder. The dimension of the input and output layers depends on different datasets. The latent vector changes correspondingly.

D. Attacks

Four attacks are considered in our experiments. These attacks have been briefly mentioned in Sec. III. More details of the attack configurations are provided next:

- ***Sign Flipping Attack***: Generates adversarial client gradient values via flipping the sign of the average sum of benign client gradients at each epoch. The updated model of a adversarial client i is $w_i = -rw_i$, where $r > 0$. We assume that w_i is a fixed vector without randomness.
- ***Additive Gaussian Noise Attack*** : The adversarial client adds Gaussian random noise which are independent random samples following a Gaussian distribution to its model update w_i. We assume that these independent Gaussian variables have the same mean yet different variances under attacks.
- ***A Little is Enough Attack***: This attack aims to prevent model convergence by hiding small perturbations within the variance of model gradients from benign clients. The adversary works by controlling the adversarial clients to send gradients computed from $Mean$.
- ***Zero Gradient Attack***: Performs sign flipping in an attempt to modify the aggregated model to record a zero value at each epoch. The zero gradient attack is a special case of sign flipping attack, where $w_i = -rw_i$ and $r = 0$.

E. Benchmark Defense Schemes

No-attack: No-attack indicates the model is trained by model updates from 100% benign clients, which provides the best model performance.

FedAvg: is a baseline of FL that allows local clients to perform more than one batch update on the local dataset and exchanges the updated weights, instead of gradients [1].

GeoMed: Instead of taking the weighted average of the local model updates as performed using FedAvg, the GeoMed method generates a global model update using the geometric median (GeoMed) of the local model updates (including adversarial updates), which may not be among local model updates [12].

Krum: Differing from GeoMed, the Krum method generates a global model update using one of the local updates, which minimizes the sum of distances to its closest neighbors (including adversarial ones) [8].

No-defense: Within RAFL, the no-defense method is no rule-based and VAE-based detector, and leverages a federated meta learning framework with multiple learning loops.

F. Evaluation metrics

Model accuracy: Used for evaluating classification models, defined as the percentage of correct classifications that a trained ML model achieves, calculated by dividing the number of correct predictions by the total prediction number.

Training time: The time taken by a ML model to train on a dataset and it also includes data processing.

G. Performance Evaluation

Model Accuracy: Fig. 5 shows model accuracy across two datasets for four different attacks, and for detection methods. Fig. 5 (a) demonstrates that RAFL achieved a higher accuracy than other approaches when applied to MNIST dataset. Our RAFL detector achieved an almost identical model accuracy to that of No-attack (88.7% vs. 89.6% for sign flipping attack). This result is due to RAFL detecting and removing adversarial clients by considering mean and standard deviation within the ranking matrix, and error reconstruction in VAE latent space. Other approaches are able to achieve satisfactory performance for Gaussian noise attack and little is enough attack for FedAvg (78.8%, 84.6%) and GeoMed (85%, 82.7%), respectively. However, both FedAvg and GeoMed are less effective against the sign flipping attack (54.7%, 39.5%) and zero gradient attack (41.8%, 30.4%). Within sign flipping attack and zero gradient attack, adversarial clients attempt to modify the geometric center of all uploaded model updates from clients, resulting in poor model performance. FedAvg calculates the average of all client model updates, resulting in model performance loss. Krum exhibits a lower model accuracy compared with FedAvg, GeoMed and RAFL. For example, under Gaussian noise attack, Krum achieves 37.75%, while FedAvg, GeoMed and RAFL achieve 78.8%, 85%, and 89.6%, respectively. The is due to Krum only selecting the most appropriate updates, however in a non-IID setting client model updates are biased, thus decreasing model accuracy.

As shown in Fig. 5, our proposed RAFL exhibits a similar trend, achieving the best performance in all settings when applied to Cifar10. Since the Cifar10 dataset and model architecture of the CNN model are more complex, the model accuracy is lower than MNIST. For instance, under sign flipping attacks, RAFL achieved 55.5% on Cifar10 dataset. FedAvg, GeoMed and Krum all failed when exposed to sign flipping attacks, little is enough attack, Gaussian noise attack, and zero gradient attack. The exception is GeoMed exposed to a Gaussian noise attack with model accuracy (52.75%) close

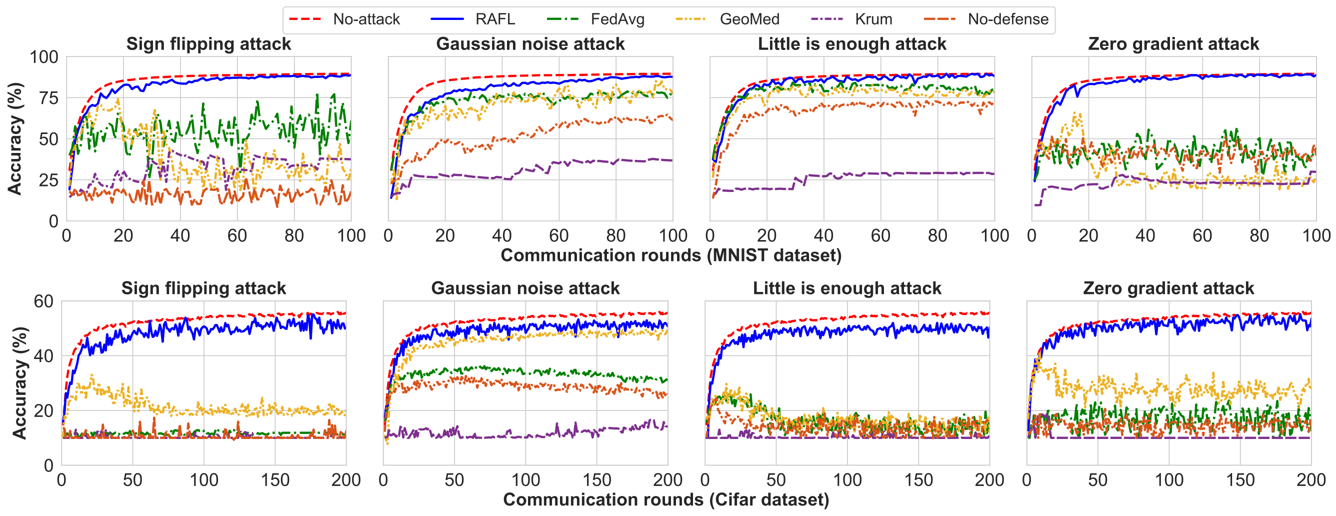

Fig. 5. Comparison of the benchmark defense schemes and RAFL using different datasets.

TABLE I
DIFFERENT NUMBER OF ADVERSARIAL CLIENTS (SIGN FLIPPING ATTACK)

Methods	10%	20%	30%	40%
RAFL	88.45%	88.52%	88.94%	89.52%
GeoMed	87.93%	85.45%	73.27%	47.47%
FedAvg	80.12%	73.26%	69.94%	48.72%
Krum	26.92%	37.5%	29.79%	30.75%

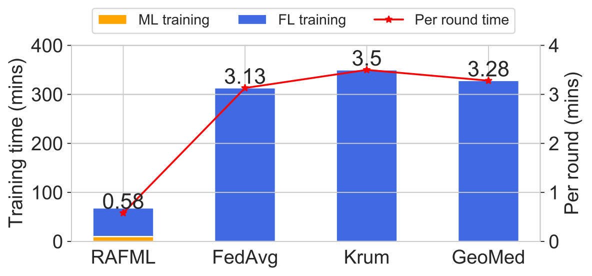

Fig. 6. Training time for 100 communication rounds

to RAFL's (53.3%). This result is due that the Gaussian noise attack adds a uniform noise $u \in U(-0.5, 1.5)$. GeoMed can utilize the geometric median to avoid such noise.

Adversarial Clients Number Effects: Table I shows the impact of the number of adversarial clients on RAFL and other benchmark defense schemes. We vary the fraction of adversarial clients from 10% to 40%, because we assume that benign clients are the majority of clients. This experiment trained a LR model on MNIST dataset. As shown in Table I, the model accuracy of GeoMed and Fedavg reduces with an increasing number of adversarial clients. For example, under GeoMed aggregation method, when 10% of clients are adversarial clients, model accuracy is 87.93%. While model accuracy is 47.47% when adversarial clients increase to 40%. It is because these two methods are calculated as a weighted average and geometric median of the resulting model from the clients, respectively. More adversarial clients produce worse results. In contrast to GeoMed and FedAvg, Krum only uses one of the local updates to generate a global model resulting in its model accuracy being less relevant to the number of adversarial clients. Our proposed RAFL is also not affected by the proportion of adversarial clients. Differing from robust learning methods that tolerate a limited proportion of adversarial clients, RFAL is a detection-based method to detect and remove adversarial clients. The absolute number has little effect on final model accuracy. As shown in Table I, when 10% of clients are adversarial, model accuracy can achieve 88.45% after 100 communication rounds. When 40 clients are adversarial, a similar result can be achieved, obtaining 89.52% model accuracy. Thus, Table I demonstrates our method effectively detects adversarial clients without assuming any number of adversarial clients, as we rank and cluster participating clients parameters and analyze their latent space.

Training Time: RAFL operates with reduced training time in comparison to other benchmark defense schemes. The total training time of RAFL consists of two parts: ML detector training time and FL training time. As shown in Fig. 6, the ML detector training time is about 8 mins and the FL training time is around 58 mins for the whole training process in 100 communication rounds. The average time of per communication round is 0.58 minutes. In contrast, FedAvg, Krum and GeoMed do not require ML detector training time. They only need FL training, but they take a longer time. FedAvg running 100 communication rounds spent 313 minutes, Krum used 350 mins and GeoMed took 328 minutes. Thus, although our proposed method needs one more step to train an ML model, it requires less training time, speeding up the training process and adaptation to new environments.

Rule-based Detector: In our proposed method, we utilized a rule-based method to generate a clean dataset to online train a

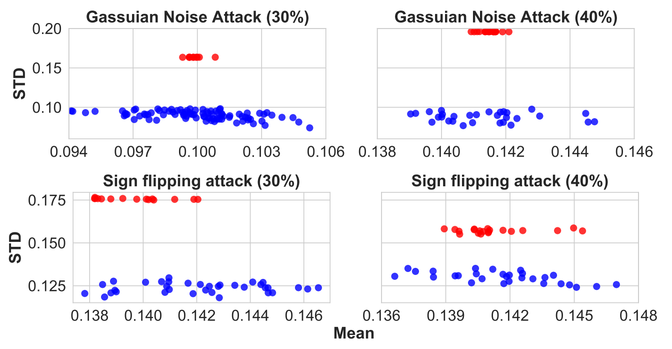

Fig. 7. Rule based detector (Red = adversarial clients, Blue = benign clients)

VAE detection model. Fig. 7 shows the scatter plots of (e_I, s_i) for 50 clients under Gaussian noise attack and sign flipping attack with 30% and 40% adversarial clients on MNIST dataset. It illustrates the difference between benign and adversarial clients in terms of the mean and the standard deviation of gradient ranking. In Fig. 7, for the sign flipping attack with 40% adversarial clients, as the number of adversarial clients increases, the difficulty of distinguishing benign clients from adversarial clients increases. The difference between them is not clear in regards to mean and standard deviation.

Discussion: As shown from our results, targeted defenses that distinguish adversarial model updates produced by different datasets are increasingly necessary. When conducting no-defense experiments, no-defense mechanisms was applied to the framework. As shown in Fig. 5, model accuracy of Gaussian noise attack, little is enough attack, and zero gradient attack is higher than Krum, demonstrating that RAFL is robust-by-design. Even without considering any detection models, RAFL still provides limited defense against attacks. Based on our experiments, the proposed method obviously cannot detect all adversarial clients. For the rule-based detection method, if adversarial clients create more adaptive attacks, they may evade our detection model and degrade model performance. For VAE online detection method, distortion is unavoidable due to dimension reduction, however does not need a public dataset, protecting clients privacy and reduce security risks.

VI. Conclusion

In this paper, we have proposed a novel FL framework that is robust against adversaries and combines a rule-based detection method and an online learning-based detection method to effectively distinguish adversarial clients from benign clients. This framework also utilizes multiple learning loops to train multiple personalized detection models for different group clients. Numerical results show that our proposed RAFL outperforms the benchmark defense methods in terms of model accuracy and efficiency.

Acknowledgments

Research supported by the UKRI Trustworthy Autonomous Systems Node in Security/EPSRC Grant EP/V026763/1.

References

[1] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial intelligence and statistics (AISTATS)*. PMLR, 2017, pp. 1273–1282.
[2] D. Kawa *et al.*, "Credit risk assessment from combined bank records using federated learning," *International Research Journal of Engineering and Technology*, vol. 6, no. 4, pp. 1355–1358, 2019.
[3] J. Xu *et al.*, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, no. 1, pp. 1–19, 2021.
[4] Z. Yu *et al.*, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5341–5351, 2020.
[5] E. Bagdasaryan *et al.*, "How to backdoor federated learning," in *Proceedings of the International Conference on Artificial intelligence and statistics (AISTATS)*. PMLR, 2020, pp. 2938–2948.
[6] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, 2021.
[7] Z. Wu *et al.*, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4583–4596, 2020.
[8] P. Blanchard *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
[9] Z. Sun *et al.*, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.
[10] C. Xie *et al.*, "Generalized byzantine-tolerant sgd," *arXiv preprint arXiv:1802.10116*, 2018.
[11] D. Yin *et al.*, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 5650–5659.
[12] Y. Chen *et al.*, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.
[13] C. Xie *et al.*, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 6893–6901.
[14] S. Li *et al.*, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.
[15] Z. Zhang *et al.*, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.
[16] Z. Gu and Y. Yang, "Detecting malicious model updates from federated learning on conditional variational autoencoder," in *International Parallel and Distributed Processing Symposium (IPDPS)*, 2021, pp. 671–680.
[17] L. Li *et al.*, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 33, no. 01, 2019, pp. 1544–1551.
[18] S. Shen *et al.*, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Annual Conference on Computer Security Applications (ACSAC)*, 2016, pp. 508–519.
[19] M. Aledhari *et al.*, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, 2020.
[20] W. Zhu *et al.*, "Mandera: Malicious node detection in federated learning via ranking," *arXiv preprint arXiv:2110.11736*, 2021.
[21] J. Kang *et al.*, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.
[22] W. Chen *et al.*, "Boosting decision-based black-box adversarial attacks with random sign flip," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 276–293.
[23] G. Baruch *et al.*, "A little is enough: Circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[24] T. Hospedales *et al.*, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
[25] C. T Dinh *et al.*, "Personalized federated learning with moreau envelopes," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 394–21 405, 2020.
[26] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 421–436.
[27] D. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[28] S. Kornblith *et al.*, "Similarity of neural network representations revisited," in *International Conference on Machine Learning (ICML)*, 2019.