FedCAM - Identifying Malicious Models in Federated Learning Environments Conditionally to Their Activation Maps

Reda Bellafqira, Gouenou Coatrieux, Mohammed Lansari, Jilo Chala

IMT Atlantique
INSERM UMR 1101 Latim
Brest, France
Email: {reda.bellafqira, gouenou.coatrieux, mohammed.lansari, chala-tura.jilo}@imt-atlantique.fr

***Abstract*—Federated Learning (FL) is a machine learning paradigm enabling collaborative model training across multiple participants and a server with decentralized data while maintaining privacy. However, FL is susceptible to adversarial poisoning attacks where some malicious participants may fail the convergence or change the behavior of the model under some conditions. This paper introduces FedCAM, a robust framework designed to detect and exclude malicious model updates in FL. FedCAM uniquely utilizes Activation Maps (AMs) from representative data samples (a trigger set), unlike other methods relying on surrogate vectors of model parameters. It leverages a conditional variational autoencoder (CVAE), conditioned on the trigger set's label class, to evaluate the reconstruction error of geometric median normalized AMs. FedCAM dynamically determines the decision threshold for each round based on CVAE's error, proving more effective in detecting poisoning compared to existing methods, as demonstrated by our experiments on the IID MNIST dataset. This approach ensures efficient detection of malicious updates in every FL round.**

***Index Terms*—Deep Learning Model, Byzantine Attack, Poisoning Attack, Auto-Encoder**

I. Introduction

Federated learning (FL) represents a solution for training a global model between remote clients while ensuring the confidentiality of their private data [1]–[3]. In this framework, a central server initiates a global model and transmits it to a group of clients for training purposes. Subsequently, the clients conduct local training of the global model on their dataset. To preserve data privacy, they employ techniques such as differential privacy [4] or homomorphic encryption [5], and then transmit the updated global model back to the central server. Thereafter, the central server aggregates these updates using methods such as FedAvg [6], KRUM [7], GeoMed [8], SCAFFOLD [9] and FedProx [10] before sending the updated global model to clients. This iterative process continues until the global model is trained.

This work was partly supported by the Inserm industrial chair CYBAILE and two French government grants managed by the Agence Nationale de la Recherche under the France 2030 program, bearing the references ANR-22-PESN-0006 (PEPR digital health TracIA project) and ANR-22-PESN-0014 (PEPR digital health SSF-ML-DH project), respectively.

Even though such an FL process ensures the privacy of the client's data, mainly because the central server does not have access to local data samples [11], its decentralized nature renders it susceptible to certain attacks, especially poisoning attacks. In these attacks, malicious clients can introduce corrupted samples during the training phase to alter in a given way the performance of the machine learning (ML) model. In this paper, our focus is directed towards two distinct types of poisoning attacks, specifically the Byzantine attack and the Targeted Model Poisoning Attack (TMPA). A Byzantine attack consists of arbitrarily modifying the updates sent by clients, to degrade the performance of the global model or even cause the learning phase to fail [12]–[14]. The main objective of a TMPA is to misclassify a pre-selected set of inputs [15] taking advantage of the lack of transparency in client updates. Thus, the development of an aggregation method capable of detecting and removing malicious updates in time remains a significant challenge to the practical implementation of FL.

A. Related Work

The domain of Byzantine-robust federated learning has attracted considerable attention. The majority of existing literature including [7], [8], [16], has focused on extending stochastic gradient descent to defend against Byzantine attacks. Nevertheless, these aggregation strategies are not efficient when confronting TMPA [17], [18].

Regarding TMPAs, several proposals make use of Variational Autoencoders (VAE), renowned for their anomaly detection capabilities. These methods use VAE reconstruction error as a probabilistic anomaly measure, offering a more objective score than traditional autoencoder-based and principal components-based anomaly detection. Specifically, Li *et al.* [17] proposed a VAE-based defense framework over local model updates, demonstrating significantly larger reconstruction errors for malicious updates compared to benign ones. Nonetheless, this approach requires VAE to be trained on normal instances to achieve good defense performance under specific malicious attacks, such as same-value attacks. This is a strong assumption in real applications, considering the

© 2024 International Federation
for Information Processing (IFIP).
ISBN: 978-3-903176-61-4

© 2024 International Federation
for Information Processing (IFIP).
ISBN: 978-3-903176-61-4

difficulty in detecting poisoning attacks by design. To enhance TMPA defense, researchers have also explored Conditional VAE. FedCVAE [18], a defense framework that does not require training to detect malicious client updates. To do so, when the central server receives local updates from clients, it computes and normalizes a surrogate vector for each client model using the Geometric Median (GM). Surrogate vectors close to the GM have values near zero, contrasting with those far from the GM. These normalized vectors are then inputted into the CVAE, conditioned on the current round number in the FL training. As the CVAE is not pre-trained, inputs near zero yield reconstruction errors close to zero, and vice versa. However, the fact that this method relies on surrogate vectors for model parameters complicates the detection of certain attacks, such as backdooring or others that slightly modify the model parameters, as we will see in Section IV.

Methods like FedGuard [19] follow a different strategy to these approaches. In FedGuard, at each round, the server distributes the global model and a CVAE the participants both train on their local data. Clients return to the server the trained global model and the CVAE decoder only. The server then used the CVAE decoder to generate synthetic datasets to audit each client's model performance directly on the global learning task. Models with low performance are excluded from the aggregation. While FedGuard shows promise, it incurs significant communication overhead and demands extensive client resources, posing scalability challenges. Additionally, clients' data privacy could be a concern, as the synthetic data generated could inadvertently reveal sensitive information.

B. Contribution

To address the above issues, we propose a novel anomaly detection framework: FedCAM. Different from the previous approaches that focus on clients' model updates or surrogate vectors, FedCAM makes use of the activation maps (AMs) of a hidden layer obtained for a given set of trigger samples [20]. These AMs are fed into a CVAE conditioned on the labels of the trigger samples, to effectively detect and reject malicious updates at the server level. Our primary hypothesis is that in FL models, a poisoning attack influences not only the model parameters but also the distribution of its AM.

Subsequently, it normalizes the AMs of all selected clients using a geometric median. Then, it evaluates the reconstruction errors using the CVAE, with the trigger set labels serving as a conditional factor. The server then employs FedAvg for aggregation, applying it to non-malicious updates. As we will see, AMs normalization is at the basis of Byzantine attack detection while TMPA identification mostly relies on CVAE and its condition, especially in the case of backdooring. In our experiments, We tested our method under Byzantine attacks and TMPAs when 10% or 30% of all the clients are attackers, a common hypothesis in FL. The results show that our methods converge rapidly and stably on an IID dataset.

The basic principles of our FedCAM framework are as follows. Before initiating the training process, the server uses the initial global model to compute the AMs of one of its hidden layers for a given subset of test images, i.e. the trigger set. Then it normalizes these AMs using Geomed (for geometric median) and trains a CVAE model conditioned by the trigger set sample labels. During each FL training round, the server calculates the geomed normalized AMs for each client-updated model using the trigger set. It next evaluates the AMs CVAE reconstruction error to decide whether a client model is benign or not. The server then employs FedAvg for aggregation, applying it to non-malicious updates. As we will see, AMs normalization is at the basis of Byzantine attack detection while TMPA identification mostly relies on CVAE and its condition, especially in the case of backdooring. In our experiments, We tested our method under Byzantine attacks and TMPAs when 10% or 30% of all the clients are attackers, a common hypothesis in FL. The results show that our methods converge rapidly and stably on an IID dataset.

The contributions of this paper are summarized as follows:

- We propose a CVAE-based detection framework named FedCAM, which can effectively detect and remove malicious model updates based on the combination of the AMs distribution of a trigger set and their labels as CVAE conditions.
- We evaluate the performance of FedCAM on IID federated datasets under Byzantine attack and TMPA.
- We compare the convergence rate of FedCAM and FedCVAE [18], and the results demonstrate the superiority of the proposed model on the backdooring attacks.

II. Backgroud

In this section, we review the basics of federated learning (FL) and autoencoders. These concepts are key ideas used in anomaly detection.

A. Federated Learning

Fig. 1 shows a common scenario of federated learning. Basically, the training process is managed by an orchestrating server S. At first, this server initializes the global model M_G to be trained and sends it to a few selected clients in the federation to be trained locally on their datasets. Clients send their updated models or in some cases only their gradients, back to the server, which aggregates them to refine the global model. A common method for this aggregation is FedAvg [21]. This process of local training and aggregation is carried out iteratively until the global model achieves the desired performance or until other pre-established criteria are met. Formally, one refines the FL process as follows. Let K be the number of clients in the federation, D_k the dataset with n_k samples of the k^{th} client C_k. The training process is conducted over a series of rounds, such that at each round t:

1) S randomly selects U clients from $\{C_k\}_{k=1..K}$ and sends them the current global model M_G^{t-1}.
2) Each selected client C_u updates M_G^{t-1} on his dataset D_u such that: $M_u^t = LocalTrain(M_G^{t-1}, D_u)$, where M_u^t is the updated local model. Next, C_u sends M_u^t to S.

© 2024 International Federation
for Information Processing (IFIP).
ISBN: 978-3-903176-61-4

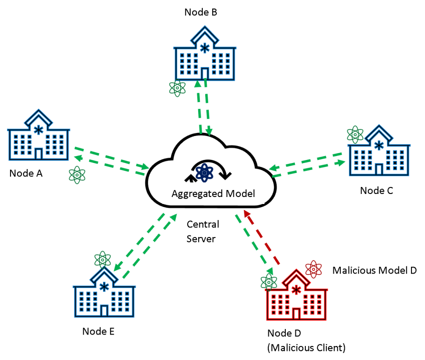


Fig. 1: Federated learning (FL) - At a training round t several clients $\{C_u\}_{u=1..U}$ update the current model M_G^{t-1} on their local dataset (D_u) and send their resulting models M_u^t to a server S. S aggregates these updates to generate an updated global model M_G^t. In the context of poisoning, a "Malicious Client" in red can send a potentially harmful model.

3) Once S received the models $\{M_u^t\}_{u=1..U}$, it aggregates them using for instance FedAvg which is a weighted sum to get the updated global model for round t such as:

$$M_G^t \leftarrow \sum_{u=1}^{U} \frac{n_u}{\sum_{u=1}^{U} n_u} M_u^t \quad (1)$$

where n_u is the cardinality of the u^{th} client samples.

This iterative process continues until the global model M_G is sufficiently trained, taking advantage of the clients' datasets D_k without centralizing them.

B. Autoencoders

One can find different variants of autoencoders. The one we're interested in is the most advanced.

1) Autoencoder (AE): Autoencoders are unsupervised and semi-supervised neural networks oriented towards efficient encoding, facilitating tasks such as dimensionality reduction, feature extraction, and data reconstruction [22], [23]. Formally and as illustrated Fig. 2(a), the architecture of an auto-encoder is first constituted of an encoder network, or equivalently an encoder function f_θ of parameters θ, that maps a given input x into a hidden representation z [24]. Its second part corresponds to a decoder network or function g of the parameter ϕ that attempts to map z to a reconstructed/approximated version of x, i.e. $\widetilde{x}$, in the original input space [25], [26]. The forward pass of an autoencoder can be summarized as follows:

$$z = f_\theta(x) \quad ; \quad \widetilde{x} = g_\phi(z) \quad (2)$$

As stated, the goal of an autoencoder is to minimize the disparity between the input x and its reconstruction $\widetilde{x}$. While several metrics can quantify this difference, the Mean Squared Error (MSE) remains prevalent:

$$MSE(x, \widetilde{x}) = \|x - \widetilde{x}\|^2 \quad (3)$$

Autoencoders have emerged as a prominent tool for anomaly detection [27]. The core idea is that anomalies can be detected based on the extent of reconstruction errors when the input data deviates from the features learned by the autoencoder. By appropriately thresholding these errors, it becomes possible to discriminate anomalies from regular data [28], [29]. However, autoencoders have a fixed-dimensional latent space, which may not be suitable for datasets with varying complexities. VAEs address this by introducing a probabilistic latent space.

2) Variational Autoencoder: Variational Autoencoders (VAEs) [30] combine the principles of autoencoders and variational inference [31]. One VAE is trained to minimize the Kullback-Leibler divergence between the latent distribution of the input data and a prior distribution $p(z)$, which is typically a standard normal distribution [31]. VAEs have gained in popularity due to their effectiveness in unsupervised learning and generative modeling compared to simple auto-encoders [26], [31]–[33]. Fig. 2(b) depicts the common VAE architecture, which can be described as follows. The encoder approximates the posterior distribution $q_\theta(z|x)$ and the decoder models the data generation process distribution with $p_\phi(x|z)$ by maximizing the evidence lower bound (ELBO) such as:

$$\mathcal{L}(x; \theta, \phi) = \mathbb{E}[\log p_\phi(x|z)] - D_{KL}(q_\theta(z|x) \| p(z)) \quad (4)$$

where, D_{KL} represents the Kullback-Leibler divergence and $p(z)$ is the prior distribution on the latent space, typically assumed to be a unit Gaussian distribution. To summarize :

- The data input x is passed through an encoding layer to produce the mean μ and standard deviation σ of a Gaussian distribution in the latent space.
- Samples from this distribution, represented by z, are drawn using the re-parameterization trick [30] (illustrated in Fig. 2(c)) and passed through a decoding layer to reconstruct the output $\tilde{x}$.

However, if the anomalies can be better characterized by adding extra information such as time, specific conditions, or data labels, Conditional VAEs usually outperform anomaly detection by VAEs.

3) Conditional Variational Autoencoders: Conditional Variational Autoencoder (CVAE) extends VAE by constraining the encoding and generative processes (see Fig. 2(c)). Given a data sample x and an associated condition y:

- The encoder approximates the posterior distribution over the latent variable z: $q_\theta(z|x, y)$ where θ denotes the encoder parameters.
- The decoder assesses the likelihood of the data based on the latent variables z and the condition y: $p_\phi(x|z, y)$ with ϕ being the decoder parameters.

The primary goal of CVAE is to maximize ELBO based on the condition y, assuming that, the latent variables typically follow a standard normal prior distribution conditioned on y: $p(z|y) = \mathcal{N}(z; 0, I)$. This can be expressed as:

$$\mathcal{L}(x, y; \theta, \phi) = \mathbb{E}[\log p_\phi(x|z, y)] - D_{KL}(q_\theta(z|x, y) \| p(z|y)) \quad (5)$$

By doing so, CVAE offers a structured approach to guide the model's generative capability based on specific conditions,

© 2024 International Federation
for Information Processing (IFIP).
ISBN: 978-3-903176-61-4

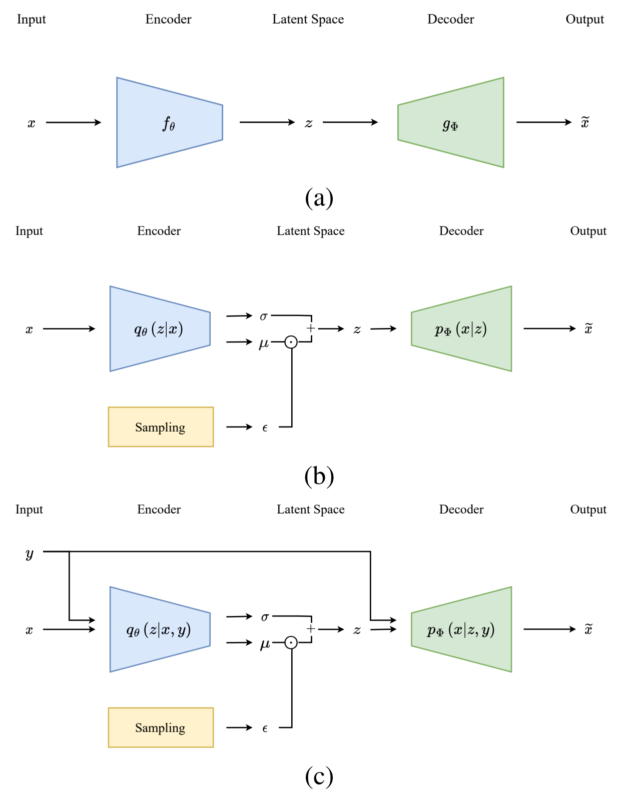


Fig. 2: Architectures: (a) Autoencoder, (b) Variational Autoencoder, (c) Conditional Variational Autoencoder.

facilitating the generation of targeted outputs. Noteworthy studies in this field include [18], [34], where the latter explores the potential of CVAE in detecting malicious update models in the FL context.

III. Proposed Method: FedCAM

FedCAM's originality lies in two main aspects. Unlike other approaches [17], [18] that rely on a surrogate vector of model parameters, FedCAM utilizes AMs from a specific layer of the model. These maps are associated with representative samples from the data distribution, forming a trigger set known only to the server.

Activation maps offer a detailed, layer-by-layer representation of how input data is processed through the model. This detailed representation helps in understanding how disturbances in model parameters affect data processing at each layer. Although slight alterations in model parameters may not always cause significant variations in the AMs, especially if the network's overall functionality remains intact, deviations that lead to abnormal data processing can be effectively revealed by these maps. This nuanced response to parameter changes renders activation maps a valuable tool for anomaly detection.

In FedCAM, during each round of FL training, the reconstruction error of a CVAE is used. This error is calculated over the distribution of activation maps from the model's participants, with the CVAE being conditioned on the label class of the samples in the trigger set. A dynamic threshold is employed to differentiate between malicious and normal updates in each round of FL.

Formally, consider a federation with an orchestrating server S and K clients: $\{C_k\}_{k=1..K}$, collaborating to build a global model M_G for a specific task. Each client C_k possesses its dataset D_k. The FedCAM process in a given round t is as follows:

1) S randomly selects U clients ($U < K$) from the federation and sends them the current global model M_G^{t-1}.
2) Each client C_u locally updates M_G^{t-1} with its dataset D_u and sends the updated model M_u^t back to S.
3) As shown in Fig.3, for each updated model M_u^t, S extracts the activation maps $\mathbf{F}_u^t$ from layer L_{target} associated with the trigger set $\mathbf{T} = (\mathbf{X_T}, \mathbf{Y_T})$.
4) S feeds the CVAE with $(\hat{\mathbf{F}}_u^t, \mathbf{Y_T})$, where $\hat{\mathbf{F}}_u^t$ is the GeoMed normalized form of $\mathbf{F}_u^t$, and computes the average mean square reconstruction error $\epsilon_u^t = \overline{\text{MSE}}(\widetilde{\mathbf{F}}_u^t, \hat{\mathbf{F}}_u^t)$. The CVAE is conditioned on the trigger sample label. S then compares ϵ_u^t to a dynamic threshold $\alpha^{\mathbf{t}}$ to determine if M_u^t is a malicious update.
5) S generates the updated global model M_G^t using FedAvg from the benign client updates.

In the following, we detail the training of our CVAE as well as the dynamic detection threshold for detecting malicious updates.

A. Training the CVAE Detector

The CVAE's objective is to distinguish between activation maps (AMs) from benign and maliciously updated models. The AMs in question are linked to a trigger set $\mathbf{T} = (\mathbf{X_T}, \mathbf{Y_T})$, known only to the server S. The CVAE training occurs before federated learning and involves the following steps:

1) S trains a model M_A with the same architecture as the global model M_G of the federation, using the trigger set $\mathbf{T}$ as input.
2) During training, S keeps a copy of the model after each epoch, resulting in $\{M_A^t\}_{t=1...ep}$ where ep is the number of epochs.
3) With $\{M_A^t\}_{t=1...ep}$ ready and using the trigger set $\mathbf{T}$, S extracts the corresponding AMs from the target layer L_{target}, denoted as $\{F_t\}_{t=1...ep}$.
4) These AMs are then normalized by S using the geometric median as follows:

$$\hat{F}_u = F_u - \text{GeoMed}(\{F_u\}_{u=1...ep}) \tag{6}$$

Let us recall that the geometric median of a set of points in a Euclidean space is a point minimizing the sum of distances to all points in the set. It is usefull to remove outliers in distribution. CVAE is trained using $(\{\hat{F}_u\}_{u=1...ep}, \mathbf{Y_T})$ as input, where $\{\hat{F}_u\}_{u=1...ep}$ are the normalized AMs and $\mathbf{Y_T}$ are the labels of the trigger set, serving as the condition for CVAE.

© 2024 International Federation
for Information Processing (IFIP).
ISBN: 978-3-903176-61-4

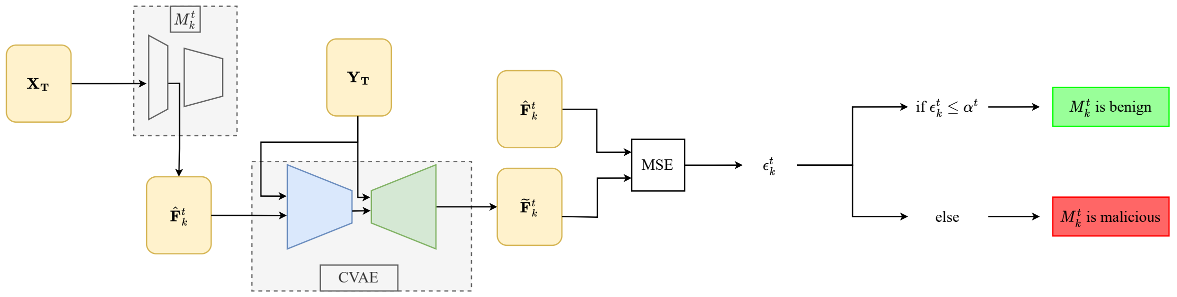


Fig. 3: Overview of FedCAM principles. Using samples from the trigger set $\mathbf{X_T}$, FedCAM extracts the activation maps $\hat{\mathbf{F}}_k^t$ of a given layer L_{target} of the updated model M_k^t provided by the client C_k at round t. From normalized $\hat{\mathbf{F}}_k^t$ and $\mathbf{Y_T}$ (the trigger set labels), CVAE produces $\widetilde{\mathbf{F}}_k^t$ and computes the average of the MSE reconstruction error: $\epsilon_k^t = \overline{\text{MSE}}(\widetilde{\mathbf{F}}_k^t, \mathbf{F}_k^t)$. If ϵ_k^t is higher than the threshold α^t, M_k^t is assumed to be a malicious model.

Since CVAE is trained with benign AMs, we anticipate that the reconstruction error of normalized AMs from malicious models will be significantly higher than from benign updates.

B. Dynamic Detection Threshold and FedCAM Algorithm

To decide whether to include an updated model M_u^t sent by the client C_u in the federation during the t^{th} round, FedCAM first normalizes the AMs using the geometric median, known for its resistance to outliers. Specifically, at each round t, the server receives the models $\{M_u^t\}_{u=1...U}$ from U clients, and computes the AMs for each client model to obtain $\{F_u^t\}_{u=1...U}$. Then, it normalizes theAMsfor each client as follows:

$$\hat{F}_u^t = F_u^t - \text{GeoMed}(\{F_u^t\}_{u=1...U}) \tag{7}$$

where $\hat{F}_u^t$ represents the normalized AMs of the u^{th} client at the t^{th} round.

Once the normalized AMs $\{\hat{F}_u^t\}$ are computed for each client, they are fed to the CVAE along with the labels $\mathbf{Y_T}$ from the trigger set $\mathbf{T}$, and the average of the normalized AMs reconstruction errors, ϵ_u^t, is computed as:

$$\epsilon_u^t = \text{MSE}(\text{CVAE}(\hat{F}_u^t, \mathbf{Y_T}), \hat{F}_u^t) \tag{8}$$

where MSE denotes the mean square error. Then, FedCAM compares this value ϵ_u^t to a dynamic threshold α_t to make a decision. In this work, α_t is defined as the mean value of the AMs reconstruction errors of all updated models at round t:

$$\alpha^t = \frac{1}{U}\sum_{u=1}^{U} \epsilon_u^t \tag{9}$$

where U is the number of clients the orchestrating server S selected at round t to update the global model M_G.

To summarize, at the FL round t, the malicious model rejection function of FedCAM operates as follows: if $\epsilon_u^t > \alpha^t$, then the model M_u^t is considered to be from a malicious client by FedCAM and will not be used in the aggregation. The global pseudo-code of FedCAM is provided in Algorithm 1.

Algorithm 1 FL Routine with FedCAM conducted by the orchestrating server

```
Require: $M_G$ the global FL model to train; $L_{target}$ the layer from which to get the AMs; $U$ number of clients; $\mathbf{T} = (\mathbf{X_T}, \mathbf{Y_T})$ the trigger set; $\mathbf{D}_k$ local dataset of the client $C_k$: CVAE trained detector.
1:  $M_G^0 \leftarrow$ Initialize($M_G$)
2:  for each round $t = 1, 2, \ldots, R$ do
3:      Randomly select $U$ clients from the federation
4:      for each client $u = 1, \ldots, U$ do
5:          $M_u^t \leftarrow$ LocalTrain($M_G^{t-1}, \mathbf{D}_u$) % $C_u$ local training
6:          $F_u^t$ = GetActivation($M_u^t, \mathbf{T}, L_{target}$) % activation maps extraction
7:          $\hat{\mathbf{F}}_u^t = \mathbf{F}_u^t -$ GeoMed($\{F_u^t\}_{u=1...U}$) % Geomed normalization
8:          $\widetilde{\mathbf{F}}_u^t$ = CVAE($\hat{\mathbf{F}}_u^t, \mathbf{Y_T}$) % CVAE reconstruction
9:          $\epsilon_u^t$ = MSE($\hat{\mathbf{F}}_u^t, \widetilde{\mathbf{F}}_u^t$) % reconstruction error
10:     end for
11:     $\alpha^t \leftarrow \frac{1}{U}\sum_{u \in C^t} \epsilon_u^t$ % dynamic detection threshold computation
12:     $S^t \leftarrow$ Selection of clients that respect $\epsilon_u^t < \alpha^t$
13:     $M_G^t \leftarrow \sum_{v \in S^t} \frac{n_v}{\sum_{v \in S^t} n_v} M_v^t$ % FedAvg aggregation, with "$n_v$" is the cardinality of the $v^{th}$ client samples.
14: end for
```

IV. Experimental Evaluation

A. FL classification task and FedCAM settings

In the following experiments, as other works from the literature, FedCAM was implemented taking into account an FL framework that aims to build a global model M_G for an image classification task on MNIST [35], a popular

© 2024 International Federation
for Information Processing (IFIP).
ISBN: 978-3-903176-61-4

public benchmark dataset for handwritten digit classification. MNIST consists of 60,000 training and 10,000 testing 28×28 pixel images, distributed among 10 distinct classes. Datasets were uniformly distributed across the customers to respect the assumption of independent and identically distributed (IID) datasets. Thus each client C_k has a local dataset D_k whose distribution is similar to that of the other clients. The code for reproducing main experiments is available at https://github.com/Bellafqira/FedCAM_.

The architecture of M_G, is as follows: three fully-connected (FC) layers with 256, 128, and 10 output neurons (ON), respectively. Relu is used as an activation function in all layers. The target Layer L_{target} from which FedCAM will extract the activation maps corresponds to the second-to-last layer. The federation we considered is constituted of 1000 clients, and $U = 50$ of them are selected randomly at each round t. The trigger set $\mathbf{T} = (\mathbf{X_T}, \mathbf{Y_T})$ is constituted of 100 images from the test set (The test set is considered public and accessible to the server in the context of FL). Regarding CVAE, its architecture is such as an FC layer with 100 ON, a latent space with 8 ON, and an FC layer with 100 ON as a decoder.

B. Considered attacks with their settings

FedCAM aims at detecting and removing byzantine and TMPA attacks. The objective of Byzantine attacks, also known as untargeted attacks, is to either prevent model convergence or to force the model to converge to incorrect values [36]. Three attacks have been tested and parameterized as in the literature [18]:

- Sign-Flipping Attacks: in which several clients reverse the sign of their model weight updates, with an amplifying factor [18]. In our tests, we applied a -1 factor to the model updates.
- Additive Noise Attack: in which several clients add Gaussian noise to their weight updates to degrade the performance of the global model [18], [37]. In our tests, a centered Gaussian distribution, $\mathcal{N}(0, 1)$, with a standard deviation of 1 was used.
- Same Value Attack: in which malicious clients consistently send the same update values at each FL round. In our experiments, malicious model parameters will take the value 0.

On their side, TMPAs subtly undermine the global FL model integrity without noticeably reducing its overall performance on its primary tasks [17], [38], [39]. We have tested the following attack:

- In a backdoor attack, a ”pattern” is added to input data. During inference, if the model receives an image from a certain class with this pattern, it misclassifies it into a specific target class [38]. Our attack implementation includes embedding a 10×10 pixel white square into images, serving as a distinct pattern, and altering labels from ”8” to ”3” for originally classified images.

C. Experimental Results

In this section, we compare the performance of our proposal, FedCAM, to FedCVAE [18], the most efficient approach, against the attacks depicted above considering, a percentage of attackers in the federation ranging from 10% to 30%. We give in Fig. 4 (a) and (b), the accuracy of the global FL models FedCAM and FedCVAE obtained under the same additive noise Attacks as well as the accuracy of the global FL model without attack and defense (baseline). As can be seen, the performance of both FedCAM and FedCVAE is very close to that of the global FL model without attack in the case of 10% of attackers. As the noise level increased to 30%, there was a very slight divergence from baseline for both defenses, even if the number of FL rounds increased. Anyway, the close similarity in performance of FedCAM and FedCVAE suggests that these mechanisms have comparable capabilities against noise attacks.

This is not the case for the same value attack. As it can be seen from Fig. 4 (c) and (d) FedCAM performs well for both 10% and 30%. FedCAM's global model accuracy remains stable and close to the baseline without attack. This is due to the AM, which can detect even slight modifications in the model parameters. Regarding the sign-flipping attack, whatever the percentage of attackers, FedCAM and FedCVAE have similar robustness and the same performance as the baseline model on IID data (see Fig. 4 (e) and (f)). To sum up on byzantine attacks, compared to FedCVAE, FedCAM offers better performance by being able to neutralize malicious update influence even for the same value attack.

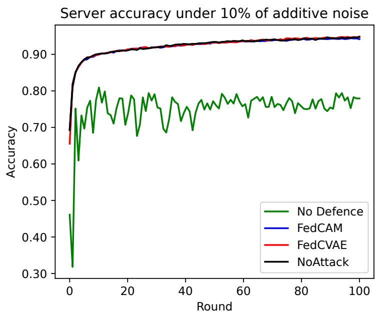
The TMPA we tested is the backdooring attack. As it can be seen from Fig. 5, for two intensities of attack, i.e. 10% and 30% of clients that agreed to misbehave, FedCAM preserves the accuracy of the global model with results comparable to the baseline (accuracy $\sim$ 90%). This is not the case for FedCVAE whose performance is unstable or more vulnerable. To go further, we also analyzed in Fig. 5 (b) and (f) the accuracy of the attack. We can see that FedCAM neutralizes the attack as soon as FL training starts, whereas FedCVAE fails to do so, especially as the number of attackers increases. If we go into more details, looking at how many attackers succeed in going through the defense each round, we can see from Fig. 5 (c), (g) for FedCAM and (d), (h) for FedCVAE that FedCAM is robust to the backdooring attack and that it outperforms FedCVAE whatever the attack intensity.

FedCAM's computational overhead compared to standard federated learning without defense is limited to the server-side. It involves three key calculations: activation maps computation with a complexity of $k \times p$ for k clients and p data points, an equal number of CVAE feedforward operations for reconstruction error calculation, and the geometric median computation often solved in a polynomial time using iterative methods.

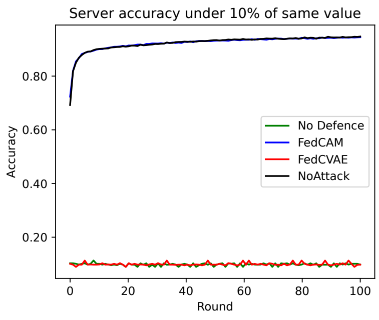
V. Conclusion & Future Work

In this work, we introduced FedCAM, a novel federated learning approach employing able to identify malicious client updates effectively. FedCAM's uniqueness stems from using

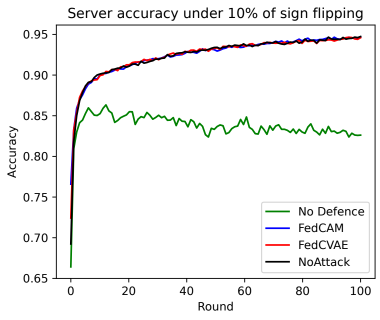
© 2024 International Federation
for Information Processing (IFIP).
ISBN: 978-3-903176-61-4



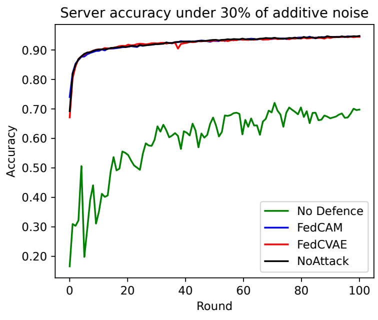
(a)



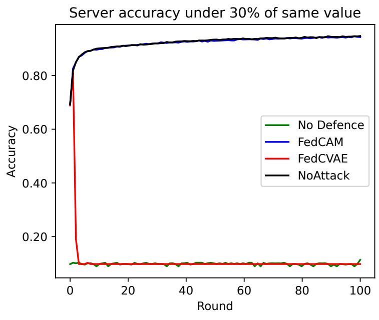
(c)



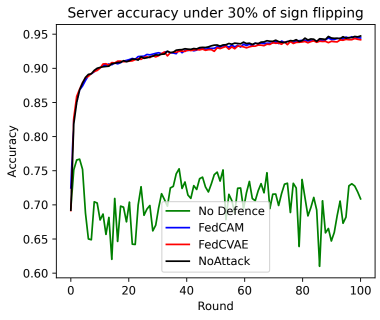
(e)



(b)

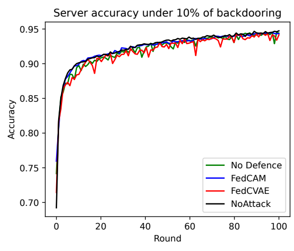


(d)

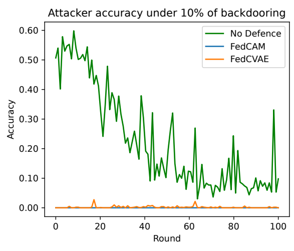


(f)

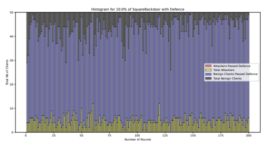
Fig. 4: FedCAM and FedCVAE Comparison Under Additive Noise (a) (b), Same Value (c) (d) and Sign-flipping (e) (f).



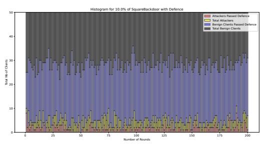
(a)



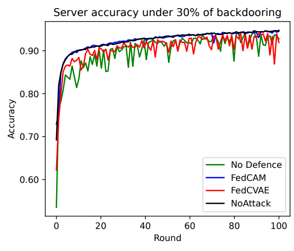
(b)



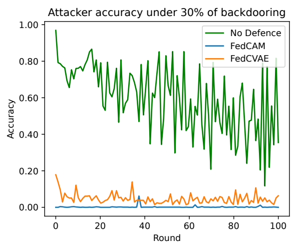
(c)



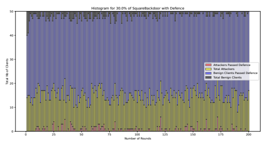
(d)



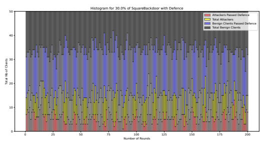
(e)



(f)



(g)



(h)

Fig. 5: FedCAM and FedCVAE Comparison Under Backdoor Attack. (c) and (g) are for FedCAM. (d) and (h) are for FedCVAE

© 2024 International Federation
for Information Processing (IFIP).
ISBN: 978-3-903176-61-4

the CVAE reconstruction error of activation maps (AM) obtained from a given trigger set; CVAE conditioned by the trigger set sample labels; and a dynamic threshold for detecting models from malevolent clients. Our experiments demonstrate FedCAM's capability in mitigating various poisoning and byzantine attacks, thus enhancing the security and convergence of global classification models under IID data distributions. Future work will focus on adapting FedCAM to non-IID distributions and applying it to broader machine learning domains, such as natural language processing and image segmentation. Further, we plan to incorporate security enhancements like differential privacy and investigate more sophisticated aggregation techniques, areas that are not extensively covered in current research.

References

[1] C. Xu, Y. Qu, Y. Xiang, and L. Gao, "Asynchronous federated learning on heterogeneous devices: A survey," *Computer Science Review*, vol. 50, p. 100595, 2023.

[2] D. Zeng, S. Liang, X. Hu, H. Wang, and Z. Xu, "Fedlab: A flexible federated learning framework." *J. Mach. Learn. Res.*, vol. 24, pp. 100–1, 2023.

[3] M. Lansari, R. Bellafqira, K. Kapusta, V. Thouvenot, O. Bettan, and G. Coatrieux, "When federated learning meets watermarking: A comprehensive overview of techniques for intellectual property protection," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1382–1406, 2023.

[4] K. Wei, J. Li, C. Ma, M. Ding, W. Chen, J. Wu, M. Tao, and H. V. Poor, "Personalized federated learning with differential privacy and convergence guarantee," *IEEE Transactions on Information Forensics and Security*, 2023.

[5] Z. Yang, Y. Chen, H. Huangfu, M. Ran, H. Wang, X. Li, and Y. Zhang, "Dynamic corrected split federated learning with homomorphic encryption for u-shaped medical image networks," *IEEE Journal of Biomedical and Health Informatics*, 2023.

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

[7] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Byzantine-tolerant machine learning," *arXiv preprint arXiv:1703.02757*, 2017.

[8] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

[9] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*. PMLR, 2020, pp. 5132–5143.

[10] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[11] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

[12] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[13] B. Zhu, L. Wang, Q. Pang, S. Wang, J. Jiao, D. Song, and M. I. Jordan, "Byzantine-robust federated learning with optimal statistical rates," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2023, pp. 3151–3178.

[14] H. Kasyap and S. Tripathy, "Privacy-preserving and byzantine-robust federated learning framework using permissioned blockchain," *Expert Systems with Applications*, p. 122210, 2023.

[15] P. Manoharan, R. Walia, C. Iwendi, T. A. Ahanger, S. Suganthi, M. Kamruzzaman, S. Bourouis, W. Alhakami, and M. Hamdi, "Svm-based generative adverserial networks for federated learning and edge computing attack model and outpoising," *Expert Systems*, vol. 40, no. 5, p. e13072, 2023.

[16] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[17] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[18] Z. Gu and Y. Yang, "Detecting malicious model updates from federated learning on conditional variational autoencoder," in *2021 IEEE international parallel and distributed processing symposium*. IEEE, 2021, pp. 671–680.

[19] M. Chelli, C. Prigent, R. Schubotz, A. Costan, G. Antoniu, L. Cudennec, and P. Slusallek, "Fedguard: Selective parameter aggregation for poisoning attack mitigation in federated learning," in *IEEE Cluster 2023-IEEE International Conference on Cluster Computing*, 2023.

[20] R. Bellafqira and G. Coatrieux, "Diction: Dynamic robust white box watermarking scheme," *arXiv preprint arXiv:2210.15745*, 2022.

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

[22] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[24] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[25] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN computer science*, vol. 2, no. 3, p. 160, 2021.

[26] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings, 2012, pp. 37–49.

[27] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM computing surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[28] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 665–674.

[29] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, and F. Sabrina, "Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset," *IEEE Access*, vol. 9, pp. 140 136–140 146, 2021.

[30] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[31] C. DOERSCH, "Tutorial on variational autoencoders," *stat*, vol. 1050, p. 3, 2021.

[32] S. Zhang, F. Ye, B. Wang, and T. G. Habetler, "Semi-supervised learning of bearing anomaly detection via deep variational autoencoders," *arXiv preprint arXiv:1912.01096*, 2019.

[33] D. P. Kingma, M. Welling *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[34] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, 2015.

[35] T. Keerthi *et al.*, "Mnist handwritten digit recognition using machine learning," in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering*. IEEE, 2022, pp. 768–772.

[36] J. Shi, W. Wan, S. Hu, J. Lu, and L. Y. Zhang, "Challenges and approaches for mitigating byzantine attacks in federated learning," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2022, pp. 139–146.

[37] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[38] E. M. Anass, C. Gouenou, and B. Reda, "Poisoning-attack detection using an auto-encoder for deep learning models," in *International Conference on Digital Forensics and Cyber Crime*. Springer, 2022, pp. 368–384.

[39] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

© 2024 International Federation for Information Processing (IFIP).
ISBN: 978-3-903176-61-4