CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Journal Paper

Exploring Visual Explanations for Defending Federated Learning against Poisoning Attacks: Enhancing LayerCAM with Autoencoders

Jingjing Zheng*

Xin Yuan

Kai Li*

Wei Ni*

Eduardo Tovar*

Jon Crowcroft

*CISTER Research Centre

CISTER-TR-251102

2025

Exploring Visual Explanations for Defending Federated Learning against Poisoning Attacks: Enhancing LayerCAM with Autoencoders

Jingjing Zheng*, Xin Yuan, Kai Li*, Wei Ni*, Eduardo Tovar*, Jon Crowcroft

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: zheng@isep.ipp.pt, xin.yuan@data61.csiro.au, kai@isep.ipp.pt, wei.ni@cister-labs.pt, emt@isep.ipp.pt, Jon.Crowcroft@cl.cam.ac.uk

https://www.cister-labs.pt

Abstract

Recent attacks on federated learning (FL) can introduce malicious model updates that can circumvent widely adopted Euclidean distance-based detection methods. This paper proposes a novel defense strategy, referred to as LayerCAM-AE, designed to counteract model poisoning in federated learning. The LayerCAM-AE puts forth a new Layer Class Activation Mapping (LayerCAM) integrated with an autoencoder (AE), significantly enhancing detection capabilities. Specifically, LayerCAM-AE generates a heat map for each local model update, which is then transformed into a more compact visual explanation. The autoencoder processes the LayerCAM heat maps from the local model updates, improving their distinctiveness and increasing the accuracy in spotting anomalous maps and malicious local models. To mitigate the risk of misclassifications in LayerCAM-AE, a voting algorithm is developed, where a local model update is flagged as malicious if its heat maps are consistently suspicious over several communication rounds. Extensive tests on the SVHN and CIFAR-100 datasets are performed under both Independent and Identically Distributed (IID) and non-IID settings in comparison with the state-of-the-art ResNet-50 and REGNETY-800MF defense models. The experimental results show that LayerCAM-AE increases detection rates (Recall: 1.0, Precision: 1.0, FPR: 0.0, Accuracy: 1.0, F1 score: 1.0, AUC: 1.0) and the test accuracy of FL, surpassing both the ResNet-50 and REGNETY-800MF. Our code is available at: https://github.com/jjzgeeks/LayerCAM-AE.

Exploring Visual Explanations for Defending Federated Learning against Poisoning Attacks: Enhancing LayerCAM with Autoencoders

JINGJING ZHENG, Real-Time and Embedded Computing Systems Research Centre (CISTER), Porto, Portugal
XIN YUAN, Commonwealth Scientific and Industrial Research Organization (CSIRO), Sydney, Australia
KAI LI*, University of Cambridge, Cambridge, U.K., and CISTER Research Centre, Porto, Portugal
WEI NI, Commonwealth Scientific and Industrial Research Organization (CSIRO), Sydney, Australia
EDUARDO TOVAR, Real-Time and Embedded Computing Systems Research Centre (CISTER), Porto, Portugal
JON CROWCROFT, University of Cambridge, Cambridge, U.K.

Recent attacks on federated learning (FL) can introduce malicious model updates that can circumvent widely adopted Euclidean distance-based detection methods. This paper proposes a novel defense strategy, referred to as LayerCAM-AE, designed to counteract model poisoning in federated learning. The LayerCAM-AE puts forth a new Layer Class Activation Mapping (LayerCAM) integrated with an autoencoder (AE), significantly enhancing detection capabilities. Specifically, LayerCAM-AE generates a heat map for each local model update, which is then transformed into a more compact visual explanation. The autoencoder processes the LayerCAM heat maps from the local model updates, improving their distinctiveness and increasing the accuracy in spotting anomalous maps and malicious local models. To mitigate the risk of misclassifications in LayerCAM-AE, a voting algorithm is developed, where a local model update is flagged as malicious if its heat maps are consistently suspicious over several communication rounds. Extensive tests on the SVHN and CIFAR-100 datasets are performed under both Independent and Identically Distributed (IID) and non-IID settings in comparison with the state-of-the-art ResNet-50 and REGNETY-800MF defense models. The experimental results show that LayerCAM-AE increases detection rates (Recall: 1.0, Precision: 1.0, FPR: 0.0, Accuracy: 1.0, F1 score: 1.0, AUC: 1.0) and the test accuracy of FL, surpassing both the ResNet-50 and REGNETY-800MF. Our code is available at: https://github.com/jjzgeeks/LayerCAM-AE

CCS Concepts: • **Security and privacy Code → Social aspects of security and privacy**; • **Computer systems organization →** *Neural networks.*

Additional Key Words and Phrases: Federated learning, poisoning attacks, class activation mapping, autoencoder

ACM Reference Format:
Jingjing Zheng, Xin Yuan, Kai Li, Wei Ni, Eduardo Tovar, and Jon Crowcroft. 2025. Exploring Visual Explanations for Defending Federated Learning against Poisoning Attacks: Enhancing LayerCAM with Autoencoders. 1, 1 (November 2025), 29 pages. https://doi.org/XXXXXXX.XXXXXXX

*Corresponding author

Authors' Contact Information: Jingjing Zheng, zheng@isep.ipp.pt, Real-Time and Embedded Computing Systems Research Centre (CISTER), Porto, Porto, Portugal; Xin Yuan, Commonwealth Scientific and Industrial Research Organization (CSIRO), Sydney, NSW, Sydney, Australia, xin.yuan@data61.csiro.au; Kai Li, University of Cambridge, Cambridge, Cambridge, U.K., and CISTER Research Centre, Porto, Portugal, kaili@ieee.org; Wei Ni, Commonwealth Scientific and Industrial Research Organization (CSIRO), Sydney, NSW, Sydney, Australia, wei.ni@data61.csiro.au; Eduardo Tovar, Real-Time and Embedded Computing Systems Research Centre (CISTER), Porto, Porto, Portugal, emt@isep.ipp.pt; Jon Crowcroft, University of Cambridge, Cambridge, Cambridge, U.K., jon.crowcroft@cl.cam.ac.uk.

Manuscript submitted to ACM

1 Introduction

Federated learning (FL) has recently emerged as a promising distributed machine learning paradigm, where multiple users collaboratively train a shared machine learning model under the orchestration of a server, while retaining the training data within individual users to mitigate the risk of data privacy leakage [39]. In FL, users iteratively train their local models (i.e., weight parameters or gradients) on their private data, and send the local models instead of the raw private data to a server for aggregation. The server utilizes the local models to compute a global model that is sent back to the users for updating their local models [18]. This process is repeated in multiple communication rounds until a desirable accuracy of image classification or convergence is achieved. This significantly alleviates the leakage of private data from the source.

While FL is able to enhance user data privacy protection, the distributed nature of FL is insufficient to ensure every local model update is benign. An attacker-controlled malicious user can potentially launch model poisoning attacks by directly manipulating local model parameters and propagating it to the global model, resulting in a corruption of the FL global model [3, 7]. To defend against model poisoning attacks on FL, existing distance-based defense mechanisms, e.g., Euclidean distance [1, 32] or cosine similarity [2], have been developed to filter out suspicious or unreliable local models before the server aggregation. However, these defense measures confront a few issues. On the one hand, excessive deletion of local model updates and/or expensive analysis of high-dimensional local model updates incur inefficiency and degradation of model quality to a great extent. On the other hand, the attacker, by eavesdropping on the benign local models to craft a malicious local model update that closely resembles benign local models, can circumvent existing defense countermeasures such as, Krum [1], Trim-mean [32], and Geometric Median [21].

In this paper, we propose a novel malicious-user detection method, called LayerCAM-AE, that identifies and filters malicious local model updates of FL from the server side. Since Layer Class Activation Mapping (LayerCAM) can visualize and interpret off-the-shelf deep neural networks (DNNs) by highlighting the regions of an input image or feature map that contribute the most to the model's predictions, LayerCAM, a visual explanation technique, is leveraged by LayerCAM-AE to discriminate a heat map for every local model update before server aggregation. LayerCAM-AE not only maps the high-dimensional local model update into a low-dimensional, reliable and precise fine-grained heat map, but also indirectly visualizes the local model update.

To further eliminate the potential errors of LayerCAM-assisted malicious user identification, i.e., a capable attacker utilizes Graph Autoencoder (GAE) to reconstruct a malicious local model that can capture the correlation of benign local model updates [15] and may evade state-of-the-art defense mechanisms, an autoencoder is embedded in LayerCAM-AE to remap the LayerCAM heat maps to accentuate the hidden features of the heat maps. This improves the distinguishability of the heat maps and the success rate of discerning anomalous heat maps and malicious local model updates.

The contributions of this paper are summarized as follows.

- We propose a new defense strategy for model poisoning attacks on FL, leveraging extended LayerCAM and autoencoder to effectively detect inconspicuous manipulations.
- The state-of-the-art LayerCAM, a visual explanation technique, tailored developed to transform the high-dimensional, indistinct local model updates in FL into low-dimensional, visually interpretable heat maps.
- An autoencoder is incorporated into the proposed LayerCAM-AE, refining the LayerCAM-generated heat maps to highlight their latent features and enhance their distinguishability. This improvement aids in more effectively identifying anomalous heat maps and detecting malicious local models.

- We also propose a voting algorithm to consistently filter out transient malicious model updates, thereby reducing the likelihood of erroneously identifying malicious local models.

We conduct a comprehensive assessment of the proposed LayerCAM-AE framework using two public datasets, SVHN and CIFAR-100, under both IID and non-IID dataset settings. Our assessment encompasses two prominent deep learning models, i.e., ResNet-50 [9] and REGNETY-800MF [22]. Our approach offers superior detection rates and FL test accuracy compared to the state-of-the-art methods. It should be pointed out that, compared to our preliminary results in [37], this work introduces a voting mechanism design and provides a detailed definition of the attacker's model. For the experiments, we have included implementation details, such as pseudo-code and parameter settings. Additionally, we have evaluated LayerCAM-AE's detection performance on IID datasets, conducted ablation studies, performed scalability experiments, and assessed its runtime. Finally, we have included discussions on the this work and its related work.

The rest of this work is organized as follows. Section 2 presents FL for image classification with deep neural networks. In Section 3, we review the state-of-the-art model poisoning attacks that can circumvent widely adopted Euclidean distance-based detection methods. In Section 4, we detail the proposed LayerCAM-AE against model poisoning attacks. In Section 4.2, we present the performance evaluation. Section 6 discusses the literature on defense models against poisoning attacks on FL. This work is concluded in Section 7.

2 Federated Learning

In this section, we exemplify image classification to delineate the training process of FL. The notations used in the paper are summarized in Table 1.

Table 1. The meaning of the notations

Notations	Meaning
$K^{'}$	The number of attackers
D_n	The local data size of benign user n
$D_{k'}$	The claimed data size of the attacker $k^{'}$
$F_k^{'}$	The local loss function of the attacker $k^{'}$
N	The number of benign users
η_t	The learning rate of user at communication round t
L	The number of epochs of local model training
S	The total data size of the local training data reported to the server
r	The threshold of Euclidean distance between malicious local model and global model
l	The index of LayerCAM heat maps
k	The index of feature map channels.
$\mathcal{P}$	The set of benign users and attackers
θ	The parameter of encoder
ϕ	The parameter of decoder
R_l	The mean reconstruction error of heat map
$\overline{R}$	The average reconstruction error of all heat maps
δ	The threshold of the mean reconstruction error
ξ	The interval of communication rounds.
O_l^t	A buffer stores the detection results in communication round t
ϵ	The number of times LayerCAM-AE detects the user as abnormal within interval ξ.

The system is composed of a server and N benign users where $\mathcal{N} \triangleq \{1, \cdots, n, \cdots, N\}$ and n is the index of the user. Each benign user trains a DNN that is particularly well-suited for image classification. The DNN analyzes an input image (e.g., cup), extracts features (e.g., shape, handle, size, and rim) through several convolutional (CONV) layers, feeds the feature maps into fully connected (FC) layers, and assigns one of the labels (e.g., "bus", "cup", "fox" or "boy"); see Figure 1. The DNN model is learned based on the training data and is represented by the parameter vector $\mathbf{W}$ (weights and biases). The user owns D_n pairs of training data $\mathcal{D}_n = \{(\mathbf{x}_d, y_d)\}_{d=1}^{D_n}$, consisting of the feature vector $\mathbf{x}_s$ as input to the model (e.g., pixels of an image) and the corresponding scalar value y_d (e.g., the real label of the image), which is the output required by the model or the desired output of the model [29].

Let $f_d(\mathbf{W}, \mathbf{x}_d, y_d)$ denote the loss function of each training data sample d, which captures approximation errors over the input $\mathbf{x}_d$ and desired output y_d. $\mathbf{W}$ is the weight parameter of the loss function in the neural network being trained according to the FL procedure. $f_d(\mathbf{W}, \mathbf{x}_d, y_d)$ can be specified according to the machine learning models. For instance, $f_d(\mathbf{W}, \mathbf{x}_d, y_d) = \frac{1}{2}\left(y_d - \mathbf{W}^T\mathbf{x}_d\right)^2$ is used to model linear regression, or $f_d(\mathbf{W}, \mathbf{x}_d, y_d) = -y_d \log\left(\frac{1}{1+\exp(-\mathbf{W}^T\mathbf{x}_d)}\right) - (1 - y_d)\log\left(1 - \frac{1}{1+\exp(-\mathbf{W}^T\mathbf{x}_d)}\right)$ is used to model logistic regression. Here, $\mathbf{W}^T$ is the transpose of $\mathbf{W}$.

For each user n, given the dataset $\mathcal{D}_n$, the local loss function of the parameter vector $\mathbf{W}$ concerning the collection of data samples is defined as

$$F_n(\mathbf{W}) \triangleq \frac{1}{D_n}\sum_{d=1}^{\mathbf{D}_n} f_d(\mathbf{W}, \mathbf{x}_d, y_d). \tag{1}$$

Accordingly, the global loss function concerning all distributed datasets is given by

$$F(\mathbf{W}) = \sum_{n=1}^{N}\frac{D_n}{D}F_n(\mathbf{W}), \tag{2}$$

where $D = \sum_{n=1}^{N} D_n$. The goal of the training process is to find the optimal parameters $\mathbf{W}^*$ such that the global loss function $F(\mathbf{W})$ is minimized, i.e.,

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} F(\mathbf{W}). \tag{3}$$

To solve the problem in (3) while preserving the data privacy for each user, a canonical gradient-descent technique is widely used in state-of-the-art FL systems [18], which is implemented iteratively in a distributed manner, as follows.

Let T_{FL} denote the total number of communication rounds with $\mathcal{T} \triangleq \{0, \cdots, T_{FL} - 1\}$, one communication round refers to that the users download global model parameter from the server and send their update local model parameters to the server. At each communication round $t \in \mathcal{T}$, the process of the FL system contains the following three steps:

- *User selection and broadcast:* The server samples N users (without loss of generality, we assume that all N users participate in FL training) satisfying eligibility requirements. The selected users download the current global model $\mathbf{W}^t$ from the server.
- *Local model training and uploading:* Each user n randomly samples a mini-batch, i.e., $\tilde{\mathcal{D}}_n$ ($\tilde{\mathcal{D}}_n \in \mathcal{D}_n$), from the local dataset $\mathcal{D}_n$ and utilizes the downloaded global model $\mathbf{W}^t$ to perform a total of L local training epochs to update the local model, i.e.,

$$\mathbf{W}_{n,\ell+1}^t = \mathbf{W}_{n,\ell}^t - \eta_t \nabla F_n(\mathbf{W}_{n,\ell}^t, \tilde{\mathcal{D}}_n),\ \ell \in \{0, 1, \cdots, L-1\}, \tag{4}$$

  where η_t denotes the learning rate at communication round t, and $\nabla F_n(\mathbf{W}_{n,\ell}^t, \tilde{\mathcal{D}}_n)$ is the local gradient estimate over $\tilde{\mathcal{D}}_n$ at local training epoch ℓ. After L local training epochs, each user n uploads its local model update $\mathbf{W}_n^t = \mathbf{W}_{n,L}^t$ to the server.

- *Global model aggregation and update:* The server aggregates N participants' updated models $\{\mathbf{W}_n^t\}_{n=1}^N$ to obtain the updated global model as

$$\mathbf{W}^{t+1} = \sum_{n=1}^{N} \frac{D_n}{D} \mathbf{W}_n^t, \tag{5}$$

  where $\mathbf{W}^{t+1}$ is sent back to all N participants.

The above process repeats until the number of communication rounds T_{FL} is met.

3 Poisoning Attacks on FL

Model poisoning attack is a striking security threat for FL, where malicious users try to compromise the global model by injecting poisoned data into the local model or intentionally tampering with local model parameters during the training process. In our attack settings, the model poisoning attack launched by the attackers whose goal is to degrade or impair the FL performance, such as accuracy and convergence, eventually resulting in denial-of-service (DoS)-type attacks. A general model poisoning attack in [7, 24] is considered in our paper as the threat model. Specifically, the considered threat model presents a general model poisoning attack in which the attacker may appear as a legitimate participant that receives the global model at each communication round. The attacker has knowledge of the global model's architecture, parameters, and the aggregation rules (such as FedAvg), aiming to craft malicious local model updates to manipulate the benign FL process. By progressively injecting malicious parameters into the aggregation process, the attacker aims to degrade the global model's accuracy over time.

During the process, all benign local models are uploaded to the server. The attackers with $\mathcal{M} \triangleq \{1', \cdots, k', \cdots, K'\}$ keep the same DNN structure as benign users. Unaware of the ill-intentioned attackers, the server collects the local model updates of all users, including both the benign and malicious, and unconsciously creates a contaminated global model update, denoted by $\mathbf{W}_g^{t+1}$, at the t-th communication round, i.e.,

$$\mathbf{W}_g^{t+1} = \sum_{n=1}^{N} \frac{D_n}{S} \mathbf{W}_n^t + \frac{D_{k'}}{S} \mathbf{W}_{k'}^t, \tag{6}$$

where $S = \sum_{n=1}^{N} D_n + D_{k'}$ is the total size of the local training data reported to the server, and $D_{k'}$ is the claimed data size of the attacker k'.

As the server may not be aware that the benign local models and malicious local models are mixed, the server unintentionally minimizes the following global loss function:

$$\min_{\mathbf{W}_g^{t+1}} F(\mathbf{W}_g^{t+1}) = \sum_{n=1}^{N} \frac{D_n}{S} F_n(\mathbf{W}_g^{t+1}) + \frac{D_{k'}}{S} F_{k'}(\mathbf{W}_g^{t+1}), \tag{7}$$

where $F_{k'}(\cdot)$ is the local loss function of the attacker, which conforms to (1).

To impair the FL training model, the attackers aim to maximize the FL global loss function $F(\mathbf{W}_g^{t+1})$, i.e., equivalent to minimizing the training accuracy of FL, while keeping $\mathbf{W}_{k'}^t$ imperceptible by the server that Euclidean distance-based [1, 32] or similarity-based detection [2]. In other words, the attacker can measure the Euclidean distance between $\mathbf{W}_{k'}^t$ and $\mathbf{W}_g^{t+1}$, and make it smaller than a threshold r:

$$\max_{\mathbf{W}_{k'}^t} F(\mathbf{W}_g^{t+1}) \tag{8a}$$

$$\text{s.t } \mathfrak{D}\left(\mathbf{W}_{k'}^t, \mathbf{W}_g^{t+1}\right) \leq r, \forall k' \in \mathcal{M}, \tag{8b}$$

where $\mathfrak{D}(\cdot)$ is the Euclidean distance function $\mathfrak{D}(\mathbf{W}_{k'}^t, \mathbf{W}_g^{t+1}) = \|\mathbf{W}_{k'}^t - \mathbf{W}_g^{t+1}\|_2$, and r is a predetermined threshold that ensures the malicious local model update is approximate to the global model within the Euclidean distance domain to bypass existing Euclidean-based distance defense mechanisms on the server side.

Note that the solution to optimization (8) can be found in [15], and the goal of this paper is not to solve this optimization problem. We aim to defend against such attacks. In particular, the threat model described in [15] represents a specific instance of the general model poisoning attack considered in our paper, where the malicious update is generated to minimize the global model's accuracy. By exploiting the feature correlation between benign local and global models, the attacker in [15] introduces subtle perturbations to the local model updates, while the malicious model remains undetected given existing Euclidean distance-based or similarity-based defense strategies. While the threat model in [15] is a concrete example, our proposed defense model remains effective even against the broader class of general model poisoning attacks in [7, 24].

4 Proposed LayerCAM-AE against FL Poisoning Attacks

The malicious local models are mixed with benign local models and uploaded to the server by the attackers. The set of the benign users and attackers is $\mathcal{I} = \mathcal{N} \cup \mathcal{M}$ with the cardinality of $I = N + K'$.

4.1 LayerCAM-AE Architecture

On the user side, each of the K benign users utilizes the DNN designed specifically for image classification tasks, as shown in Figure 1. The DNN model extracts relevant features from an input image (e.g., a butterfly) and subsequently maps them to the corresponding classes. The architecture of a typical DNN comprises multiple layers, each with a specific function. (a) The first layer is a convolutional layer, which applies a set of filters to the input image, thereby extracting intrinsic features, such as legs, thorax, and abdomen. The output of the convolutional layer is a set of feature maps, each representing different aspects of the image. (b) The following layer is a pooling layer, which reduces the dimensionality of the feature maps while retaining essential information, such as the salient features of the butterfly wings. Various pooling methods can be employed, including max pooling or average pooling, all with the objective of downsampling the feature maps while retaining their salient features. Some DNN architectures may differ from traditional pooling operations. For example, models like SqueezeNet [12], ResNet [9], DenseNet [11], and MobileNet [10] employ alternative strategies. (c) After several convolution and pooling layers, the extracted features are fed into one or multiple fully connected layers, where the final classification is performed.

Upon receiving the local model updates from the users, the server aggregates the local models, where the benign local models can be mingled with malicious local models. We aim to design a defense countermeasure that can identify and remove malicious users. LayerCAM [13] can produce reliable and high-quality class activation maps for different layers of DNN, and improves the localization accuracy of activated regions compared to CAM and GradCAM. By using multi-scale features and hierarchical representations, LayerCAM can better locate the discriminative regions in the image that contribute to the model's prediction compared to other visual explanation methods, e.g., Class Activation Mapping [38], GradCAM [23] and Grad-CAM++ [5] [23]. Unlike GradCAM, which may suffer from reduced performance with very deep networks due to vanishing gradients, LayerCAM is less affected by network depth. This is achieved by aggregating activation maps from multiple layers, ensuring that informative features are captured at each level of the network [13].

It is worth noting that class activation maps (CAM) can not be replaced in this work. This is because CAM only works with specific CNN architectures that end with a global average pooling layer followed directly by a fully

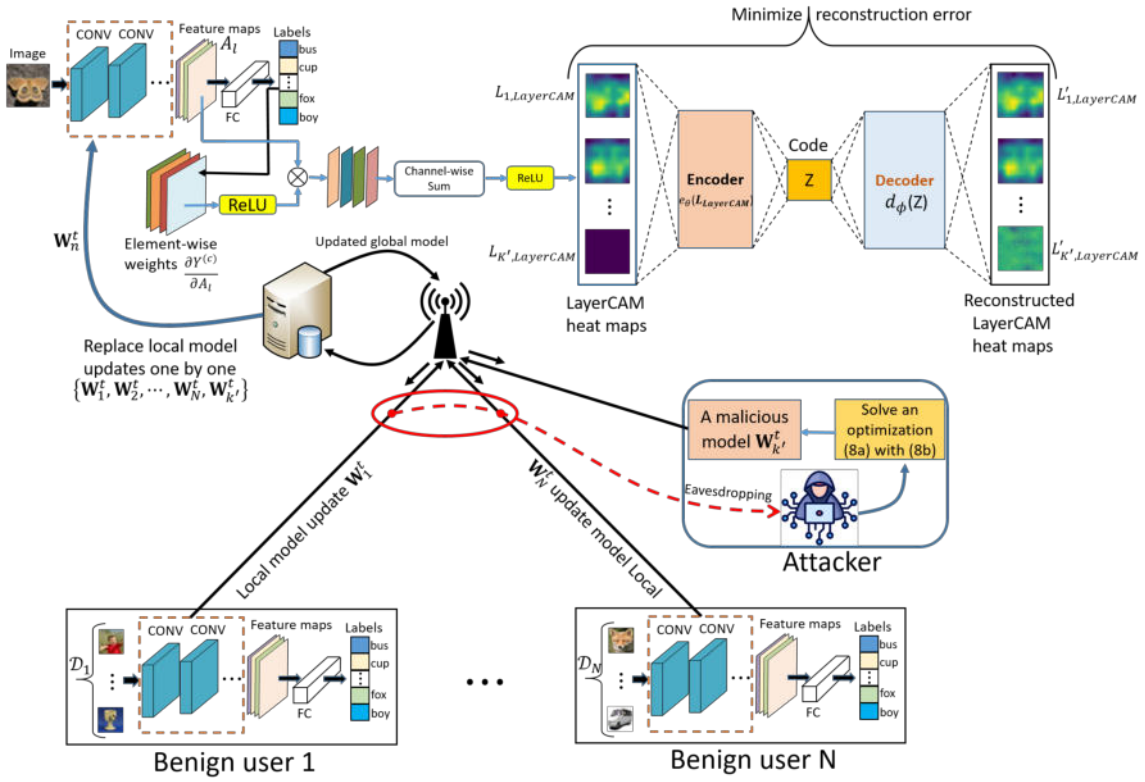


Fig. 1. Illustration of the proposed LayerCAM-AE framework, where the server arbitrarily selects an image (e.g., an image with the label "butterfly") from the global model testing dataset to create LayerCAM heat maps for every uploaded model update. These LayerCAM heat maps flow into an autoencoder. Large reconstruction errors of heat maps are judged malicious.

connected layer. In contrast, our proposed defense poisoning attack framework based on LayerCAM can work with any CNN architecture without any modifications and can generate more fine-grained heat maps to improve defense capabilities. We substantiate this point in this work using two different CNN architectures, i.e., ResNet-50 and REGNETY-800MF, which can be directly applied to our proposed framework without requiring any changes. The comprehensive experimental results are provided in Section 5.

Figure 1 illustrates our proposed LayerCAM-AE architecture that integrates LayerCAM and autoencoder, where LayerCAM generates visual heat maps for all local model updates of FL, and autoencoder identifies abnormal heat maps. The server's identification of the malicious local model updates involves two components: LayerCAM-based process and autoencoder-based process.

LayerCAM-based process. The server randomly picks an image from the global model testing dataset that incorporates all categories of the users' dataset as input and passes through the convolutional layers with weight and bias parameters that are replaced by each model update $\mathbf{W}_l, l \in \mathcal{P}$ (l is the index of the local model updates, including benign and malicious), obtains the feature map[1] A_l that has K channels. The feature maps A_l are fed into a fully connected

[1]In this paper, we focus on extracting the feature maps after the final convolution layer of deep learning model in that can capture high-level features and hold information regarding the important regions of the input image.

layer for final classification. To obtain the class discriminative localization map of $L^{(c)}_{l,LayerCAM} \in R^{B\times H}$ with width B and height H for any class c, LayerCAM first computes the gradient of the score for class c, $Y^{(c)}$ (before softmax) with respect to each spatial location (i, j) in the k-th feature map within A_l, i.e., $\frac{\partial Y^{(c)}}{\partial A^k_l(i,j)}$. Here, $k \in [1, K]$ is the index of channels. The importance of element-wise weights $\alpha^{(c)}_{l,ij}$ can be obtained through the rectified linear unit (ReLU):

$$\alpha^{(c)}_{l,ij} = ReLU\Big(\frac{\partial Y^{(c)}}{\partial A^k_l(i,j)}\Big), \forall l \in \mathcal{P}, \tag{9}$$

where $A^k_l(i, j)$ is the activation at spatial location (i, j) of the feature map A^k_l. LayerCAM further multiplies the activation value of each location in the feature map to an importance weight. The multiplications are linearly combined along the channel dimension to obtain the class activation map. The result is through ReLU operation, which is formulated as

$$L^{(c)}_{l,LayerCAM} = \text{ReLU}\Big(\sum_{k=1}^{K} \alpha^{(c)}_{l,ij} \cdot A^k_l(i, j)\Big). \tag{10}$$

Autoencoder-based process. Each LayerCAM heat map $L_{l,LayerCAM}$ with size of $H \times B$ generated from a test image and a local model update is flattened into a vector with size of $1 \times H \times B$, which is further concatenated with the vectors of other LayerCAM heat maps to form $\mathbf{L_{LayerCAM}}$ as the input of the encoder. During the autoencoder training, the encoder $e_\theta(\mathbf{L_{LayerCAM}})$ with parameter θ compresses the LayerCAM heat maps from a high-dimensional space to a low-dimensional space, i.e., $z = e_\theta(\mathbf{L_{LayerCAM}})$, also called the code or the latent space. The code learns the underlying features or representation of the LayerCAM heat maps, which are input to the decoder $d_\phi(z)$ with parameter ϕ. The decoder further reconstructs the input LayerCAM heat maps from the code, i.e., $d_\phi(z) = \mathbf{L}^{'}_{\mathbf{LayerCAM}} = d_\phi\big(e_\theta(\mathbf{L_{LayerCAM}})\big)$. After the training of (θ, ϕ), the reconstructed LayerCAM heat maps are reshaped into the same size as the original LayerCAM heat maps. To minimize the difference between the original input LayerCAM heat maps and reconstructed LayerCAM heat maps, the autoencoder loss function is defined as the mean squared error (MSE) between the encoder input LayerCAM heat maps $\mathbf{L_{LayerCAM}}$, and the decoder reconstructed LayerCAM heat maps $\mathbf{L}^{'}_{\mathbf{LayerCAM}}$, as given by

$$\begin{aligned} L(\theta,\phi) &= \min_{\theta,\phi} \frac{1}{|N| + |K'|} \|\mathbf{L_{LayerCAM}} - \mathbf{L}^{'}_{\mathbf{LayerCAM}}\|_2^2 \\ &= \min_{\theta,\phi} \frac{1}{N+K'} \sum_{l=1}^{N+K'} \|L_{l,LayerCAM} - d_\phi(e_\theta(L_{l,LayerCAM}))\|_2^2. \end{aligned} \tag{11}$$

The computational overhead of the autoencoder remains linear with respect to the number and size of the generated heat maps, ensuring that the framework remains scalable and practical for large-scale FL settings. Specifically, for the server-side implementation, the computational complexity per iteration is given by $O(I \cdot d \cdot h)$, where $I = N + K'$ represents the total number of heat maps (including those corresponding to malicious users), $d = H \times B$ denotes the dimensionality of each heat map, and h is the hidden-layer size of the autoencoder. The corresponding space complexity is $O(d \cdot h) + O(I \cdot (d + h))$, which also scales linearly with the number of heat maps and the model size of the autoencoder. Therefore, the overall design of LayerCAM-AE remains computationally efficient. The linear growth in computation and memory requirements allows the process to be parallelized on servers, reducing latency and enabling efficient handling of large-scale FL deployments. In practice, the improved detection accuracy and manageable complexity make LayerCAM-AE a viable and scalable defense mechanism for real-world FL applications.

4.2 LayerCAM-AE for Malicious Local Model Identification

Once the autoencoder completes its training process, the server computes the reconstruction errors between each reconstructed LayerCAM heat map, and its corresponding input LayerCAM heat map and obtains the mean reconstruction error, i.e., $\forall l \in \mathcal{K} \cup \mathcal{M}$,

$$R_l = \frac{\sum_{i=1}^{B} \sum_{j=1}^{H} \left| L_{l,LayerCAM}(i,j) - L'_{l,LayerCAM}(i,j) \right|}{H \times B}. \tag{12}$$

The average reconstruction error of all LayerCAM heat maps is evaluated as

$$\overline{R} = \frac{1}{N+K'} \sum_{l=1}^{N+K'} R_l. \tag{13}$$

A threshold δ is defined as

$$\delta = \overline{R} + \beta \times \sqrt{\frac{\sum_{l=1}^{N+K'} (R_l - \overline{R})^2}{N+K'}}, \tag{14}$$

where β is an empirically configured coefficient. If the mean reconstruction error for each LayerCAM heat map exceeds the threshold δ, the corresponding input of the LayerCAM heat map is considered potentially abnormal. Otherwise, it is a potential normal LayerCAM heat map because the autoencoder learns to capture variations in normal LayerCAM heat maps during training. The autoencoder can encounter difficulties in handling anomalies that do not conform to the learned patterns.

During training, the autoencoder is optimized to minimize the reconstruction error between the input and reconstructed LayerCAM heat maps, essentially learning to accurately represent normal heat map patterns. However, since attackers in FL lack access to the underlying training data of benign users, the perturbations introduced by a malicious model that significantly degrade the test accuracy of global model still manifest as distinct structural deviations in the LayerCAM-generated heat maps. These deviations indicate non-trivial alterations in the model's internal feature activation patterns. The autoencoder in LayerCAM-AE is specifically designed to capture such latent differences, enabling it to distinguish malicious from benign updates even when the per-region gradient contribution is relatively small.

Note that δ varies over communication rounds as the LayerCAM heat map changes. The detection results are stored in a buffer, as given by

$$O_l^t = \begin{cases} 1, & \text{if } R_l \leq \delta; \\ 0, & \text{if } R_l > \delta, \end{cases} \tag{15}$$

where "1" and "0" indicate a benign and a malicious local model update, respectively.

A voting mechanism is designed to further reduce the possibility of misclassifying a benign local model update. The key idea is that the server makes final decisions every fixed ξ communication rounds to determine which users are malicious. Specifically, the local model updates ϵ out of ξ communication rounds are detected as potentially malicious. They are determined malicious and removed from global aggregations. Finally, (7) is rewritten as

$$\mathbf{W}_g^{t+1} = \begin{cases} \sum_{l=1}^{N+K'} O_l^t \times \frac{D_l}{S} \times \mathbf{W}_l^t, & \text{if } t \text{ modulo } \xi \neq 0; \\ \sum_{l=1}^{N+K'} Y_l^t \times \frac{D_l}{S} \times \mathbf{W}_l^t, & \text{if } t \text{ modulo } \xi = 0, \end{cases} \tag{16}$$

where Y_l^t is given by

$$Y_l^t = \begin{cases} 1, & \text{if } \sum_{t=\Delta\xi-\xi+1}^{\Delta\xi} O_l^t < \epsilon, \Delta = 1, \cdots, \lfloor \frac{T_{FL}}{\xi} \rfloor; \\ 0, & \text{if } \sum_{t=\Delta\xi-\xi+1}^{\Delta\xi} O_l^t \geq \epsilon, \Delta = 1, \cdots, \lfloor \frac{T_{FL}}{\xi} \rfloor, \end{cases} \tag{17}$$

where $\lfloor \cdot \rfloor$ is the floor function.

The updated global model $\mathbf{W}_g^{t+1}$ is sent back to all users. The detailed pseudo-code is given in **Algorithm 1**.

Algorithm 1: The proposed LayerCAM-AE for detecting malicious local model updates on FL

Input : N - The number of benign users; T_{FL} - The total number of communication round; L - The number of local epochs; η_t - The learning rate of users; $\mathbf{W}^0$ - The initialized global model; $\mathbf{M}$ - Buffer with size $\xi \times (N + K')$; ξ - The interval of communication rounds; ϵ - Voting threshold; I_0 - An image picked form testing dataset.

Output : The final global model weight $\mathbf{W}^{T_{FL}}$

for $t = 1, \cdots, T_{FL}$ **do**
  Server broadcasts $\mathbf{W}^t$ to all users;
  $\mathbf{M} \leftarrow \mathbf{0}, \mathbf{M} \in \mathbf{R}^{\xi \times (|N+K'|)}$;
  for $n \in \mathcal{N}$ **do**
    $\mathbf{W}_n^t \leftarrow \underline{LocalUpdate(n, \mathbf{W}^t)}$;
  for $k' \in \mathcal{M}$ **do**
    $(\mathbf{W}_{k'}^t)^* \leftarrow$ Solve (8a) with constraint (8b) ;
  if t *modulo* $\xi \neq 0$ **then**
    $A_l \leftarrow \mathbf{W}_l^t$ and I_0 perform convolution operation ;
    $L_{l,LayerCAM} \leftarrow$ according to (9) and (10) ;
    $\mathbf{L}_{\mathbf{LayerCAM}} \leftarrow$ concatenate($L_{l,LayerCAM}$) ;
    $\mathbf{L}'_{\mathbf{LayerCAM}} \leftarrow$ Autoencoder($\mathbf{L}_{\mathbf{LayerCAM}}$) ;
    $O_l^t \leftarrow$ (15) ;
    Store O_l^t into $\mathbf{M}$;
  else
    $A_l \leftarrow \mathbf{W}_l^{t+1}$ and I_0 perform convolution operation ;
    $L_{l,LayerCAM} \leftarrow$ according to (9) and (10) ;
    $\mathbf{L}_{\mathbf{LayerCAM}} \leftarrow$ concatenate($L_{l,LayerCAM}$) ;
    $\mathbf{L}'_{\mathbf{LayerCAM}} \leftarrow$ Autoencoder($\mathbf{L}_{\mathbf{LayerCAM}}$) ;
    $O_l^t \leftarrow$ (15) ;
    Store O_l^t into $\mathbf{M}$;
    $Y_l^t \leftarrow$ (17) ;
  Update global model $\mathbf{W}_g^{t+1} \leftarrow$ (16).
$\underline{LocalUpdate(n, \mathbf{W}^t)}$;
for $\ell = 1, \cdots, L$ **do**
  for $b_n \in \mathcal{D}_n$ **do**
    (4).

5 Performance Evaluation

5.1 Experiment Setup

Parameter settings. We set the parameters of LayerCAM-AE as follows. The total number of communication rounds is $T_{FL} = 100$. For each communication round, 21 benign users train their local models for 25 epochs using the Adam optimizer, with a batch size of 64, a learning rate of 0.0001, and a weight decay of 0. Three attackers eavesdrop on the benign local models in each communication round. At the server, the autoencoder trains the LayerCAM heat maps for 200 epochs using the Adam optimizer, with a hidden-layer size of 128, a learning rate of 0.001, and a weight decay of 0.00001. In the voting process, the communication round interval is $\xi = 3$, and the threshold is $\epsilon = 2$. The experiments were conducted on a single Zotac GeForce RTX 4090 with 24GB GDDR6X memory.

Datasets. Four datasets, i.e., SVHN [19], CIFAR-100 [14], Tiny-Imagenet-200 [31], and CelebA [17] are used to evaluate the performance of our proposed LayerCAM-AE.

- SVHN: This dataset consists of 600,000 32×32 color images in 10 different classes. There are 73,257 training images allocated to all benign users for local model training, and 26,032 testing images are allocated to the server for the global model testing until the end of each communication round.
- CIFAR-100: This dataset contains 100 different classes of 60,000 32×32 color images. There are 50,000 training images, and 10,000 testing images are assigned to all users for training and to the server for testing.
- Tiny-Imagenet-200: This dataset contains 200 distinct object classes, each comprising 500 training images and 50 validation images, for a total of 100,000 training samples and 10,000 validation samples. All images are color images with a spatial resolution of 64×64 pixels. The training images are distributed among all users for local training, while the validation set is assigned to the server for global model evaluation.
- CelebA: This dataset consists of over 200,000 color face images of celebrity identities, each annotated with 40 attribute labels. All images are center-cropped and resized to 128×128 pixels. In our experiments, the CelebA dataset is partitioned among users in a non-IID manner based on identity, where each user receives images from a distinct subset of individuals for local training.

Benchmarks. We consider the following state-of-the-art defense schemes, i.e.,

- AUROR [25] (K-means based): AUROR employs K-means to cluster the uploaded local model updates over training rounds and discards the malicious model updates, i.e., contributions from small clusters that exceed a threshold distance are considered malicious.
- Multi-Krum [1] (Euclidean distance based): This scheme computes a score of the sum of each local model update to its neighbor's Euclidean distance. The score is the sum of its Euclidean distance from its neighbors; those with high scores as malicious model updates are excluded.
- FAA-DL [26] (SVM based): This scheme utilizes an appropriate kernel function and soft margins to estimate a nonlinear decision boundary and separate the benign and malicious local model updates.
- GradCAM-AE: In this scheme, an autoencoder is applied to identify the malicious GradCAM heat maps that are generated at the server.
- LayerCAM-Krum: In this scheme, the Krum [1] algorithm is applied to identify the abnormal LayerCAM heat maps that are generated at the server.

Table 2. Detection rates of **ResNet-50** on **IID** SVHN and CIFAR-100 with 3 attackers.

ResNet-50	IID SVHN						IID CIFAR-100					
Methods	Recall	Precision	FPR	ACC	F1 score	AUC	Recall	Precision	FPR	ACC	F1 score	AUC
AUROR	0.047	0.030	0.173	0.729	0.036	0.437	0.04	0.024	0.170	0.731	0.030	0.435
Multi-Krum	0.080	0.080	0.131	0.77	0.080	0.474	0.08	0.08	0.131	0.77	0.08	0.474
FAA-DL	0.737	0.123	0.751	0.310	0.210	0.493	0.763	0.121	0.783	0.285	0.208	0.490
GradCAM-AE	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.948	0.016	0.986	0.965	0.992
LayerCAM-Krum	0.91	0.91	0.013	0.978	0.91	0.949	0.637	0.637	0.052	0.909	0.637	0.792
LayerCAM-AE	**1.0**	**1.0**	**0.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**	**1.0**	**1.0**	**1.0**

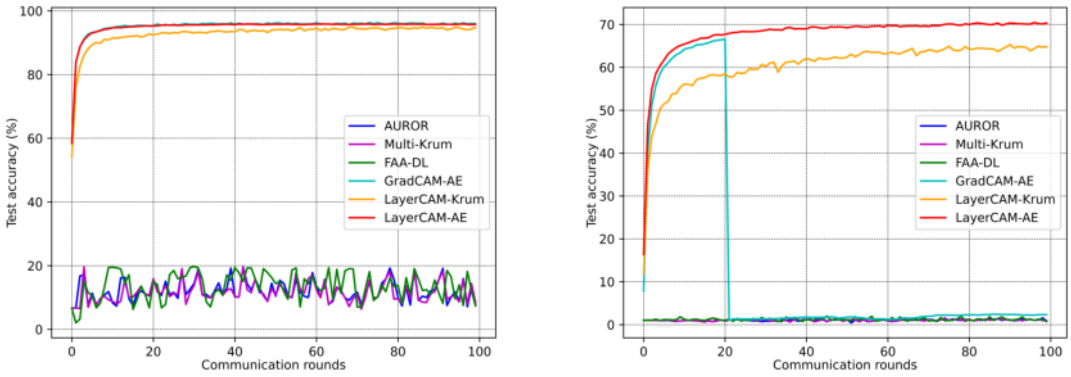
- CosDefense [30]: The server computes the cosine similarity between each user's final layer weights and those of the global model. Users with significantly higher similarity scores than the average are flagged as potentially malicious and excluded from the aggregation process during each training round.

As far as the detection rates performance of defense methods are concerned, we use following metrics to measure the detection performance of the defense approaches.

- Recall: The proportion of actual malicious users that are correctly identified by the defense model out of all malicious users.
- Precision: the proportion of correctly predicted malicious users in all predicted malicious users by the defense model.
- False positive rate (FPR): The proportion of actual benign users that are incorrectly classified as malicious by LayerCAM-AE out of all benign users.
- Accuracy (ACC): The proportion of correctly classified users in the total number of users.
- F1 score: The harmonic mean of Precision and Recall are taken as performance measures, which can be obtained by relevant calculation through a confusion matrix.
- Area under the receiver operating characteristic curve (AUC): A single scalar value between 0 and 1 that summarizes the overall ability of the defense model to discriminate between the malicious users and benign users, a larger value means the defense model is more effective at separating the malicious users and benign users.

5.2 Evaluation under IID Datasets

Fig. 2 compares FL test accuracy between the proposed LayerCAM-AE defense framework and the benchmarks, where the ResNet-50 model performs image classification on SVHN and CIFAR-100 with the IID settings. In Fig. 2a, LayerCAM-AE and GradCAM-AE can achieve the highest test accuracy of the global model as the autoencoder can accurately exclude the abnormal LayerCAM and GradCAM heat maps, respectively. As a result, more benign local model updates are involved in the FL training process. This conforms to the results of the detection rates presented in Table 2. When we replace the SVHN dataset with a more complicated CIFAR-100 dataset, GradCAM-AE is unable to detect the malicious local models in the 21st communication round, causing the malicious local models to be involved in the update of the FL global model. This is because GradCAM is unable to capture fine-grained features as the dataset becomes more complex. As FL continues to train, the disappearance of the heat maps generated by benign users becomes increasingly serious. In contrast, LayerCAM can capture more detailed features without compromising the detection of anomaly heat maps.



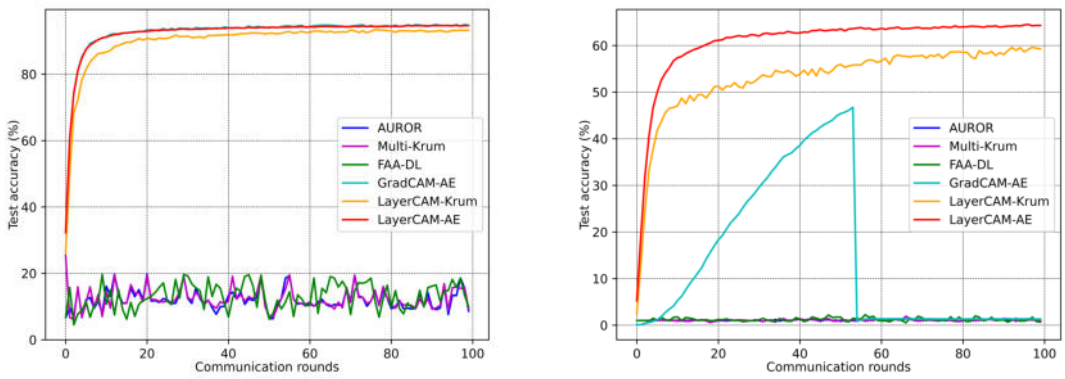
(a) SVHN (b) CIFAR-100

Fig. 2. ResNet-50 on IID SVHN and CIFAR-100.

While LayerCAM-Krum only selects an optimal local model update as the global model to train FL, it also achieves higher test accuracy (94.8% on the IID SVHN dataset and 64.8% on the IID CIFAR-100 dataset), compared to the benchmarks. The reason is that the data is IID, and the difference among the local model updates is insignificant.

Both AUROR and Multi-Krum directly detect millions, even many more, parameters of local model updates based on the Euclidean distance. The Euclidean distance between the crafted malicious local model updates and the benign ones is within the threshold designated by the server. This indicates that the malicious local model updates can elude the detection of the server and participate in the FL training process through multiple communication rounds, resulting in the global model being corrupted. Table 2 shows that although the ACCs of the Euclidean distance-based defense schemes are relatively high (0.729 and 0.77), the Precision is close to zero. In other words, they can correctly classify a few benign users but fail to identify malicious users. Similar performances are observed on the CIFAR-100 dataset, i.e., LayerCAM-AE still outperforms the benchmarks, as shown in the detection rates of ResNet-50 on IID CIFAR-100 in Table 2.

FAA-DL, a one-class SVM-based method, also directly classifies the local model updates aggregated by the server as benign and malicious. However, there are two key reasons why FAA-DL fails in the experiments. On the one hand, the local model updates have the characteristics of high-dimensional feature spaces. The curse of dimensionality can lead to data sparsity, making it challenging for FAA-DL to find a suitable margin that separates benign from malicious local model updates. On the other hand, FAA-DL is sensitive to class imbalance (21 benign local model updates and three malicious local model updates); in other words, FAA-DL biases its decision boundary towards the majority class, making it struggle to detect malicious local model updates effectively. Notwithstanding, we replace the DNN model with REGNETY-800MF. Similar performances are observed; see Fig. 3. The detection rates of REGNETY-800MF on SVHN and CIFAR-100 with the IID settings are given in Table 3. This demonstrates that LayerCAM-AE is applicable to different DNN models for detecting malicious local model updates.



(a) SVHN (b) CIFAR-100

Fig. 3. REGNETY-800MF on IID SVHN and CIFAR-100.

Table 3. Detection rates of **REGNETY-800MF** on **IID** SVHN and CIFAR-100 with 3 attackers.

REGNETY-800MF	IID SVHN						IID CIFAR-100					
Methods	Recall	Precision	FPR	ACC	F1 score	AUC	Recall	Precision	FPR	ACC	F1 score	AUC
AUROR	0.020	-	0.171	0.728	0.015	0.424	0.027	0.168	0.174	0.726	0.020	0.426
Multi-Krum	0.063	0.063	0.134	0.766	0.063	0.465	0.093	0.093	0.130	0.773	0.093	0.482
FAA-DL	0.780	0.124	0.788	0.283	0.213	0.496	0.747	0.122	00.762	0.302	0.209	0.492
GradCAM-AE	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.980	0.004	0.996	0.988	0.998
LayerCAM-Krum	0.997	0.997	0.0005	0.999	0.997	0.998	0.807	0.807	0.028	0.952	0.807	0.890
LayerCAM-AE	**1.0**	**1.0**	**0.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**	**1.0**	**1.0**	**1.0**

5.3 Performance under non-IID Datasets

Compared with the IID setting, the non-IID setting is much more challenging for the server to identify the malicious local model updates. This reason is that the local model updates tend to be more divergent under the non-IID settings; in turn, the generated heat maps on the server are more diverse. To simulate the statistical data heterogeneity in practical FL scenarios, we adopt one of the non-IID partition schemes, i.e., latent Dirichlet allocation (LDA), which are widely used in the latest literature [28, 33]. As a biased probability, the α parameter of LDA controls the distribution difference of the local training data. A larger α indicates a higher level of data heterogeneity under the non-IID setting among the local training data. Here, we set $\alpha = 0.5$.

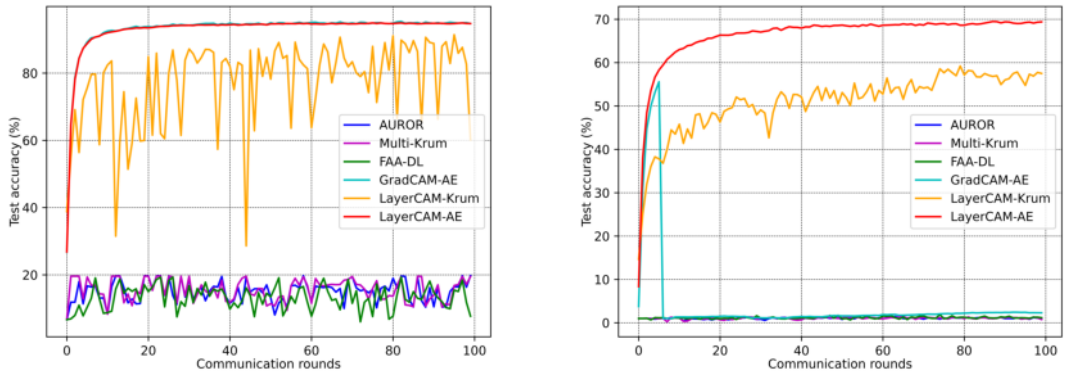
Fig. 4 plots the test accuracy of ResNet-50 on the non-IID SVHN dataset and the non-IID CIFAR-100 dataset, demonstrating the superiority of LayerCAM-AE. In Fig. 4a, LayerCAM-AE achieves the highest test accuracy under the non-IID SVHN setting. Moreover, LayerCAM-AE can quickly converge (around the tenth communication round), as it involves more benign users in FL training. This indicates that LayerCAM-AE can accurately filter malicious model updates, as can also be confirmed by the detection indicators in Table 4. Although LayerCAM-Krum can prevent malicious users from participating in FL, it sacrifices accuracy and robustness, as it selects only one local model update as the global model. The more divergent the model updates, the more diverse the heat maps. LayerCAM-Krum struggles

Table 4. Detection rates of **ResNet-50** on **non-IID** SVHN and CIFAR-100 with 3 attackers.

ResNet-50	non-IID SVHN						non-IID CIFAR-100					
Methods	Recall	Precision	FPR	ACC	F1 score	AUC	Recall	Precision	FPR	ACC	F1 score	AUC
AUROR	0.027	0.017	0.162	0.736	0.021	0.432	0.020	0.013	0.181	0.718	0.016	0.419
Multi-Krum	0.07	0.07	0.133	0.768	0.07	0.469	0.077	0.077	0.132	0.769	0.077	0.472
FAA-DL	0.720	0.122	0.721	0.334	0.208	0.50	0.703	0.128	0.680	0.368	0.215	0.512
GradCAM-AE	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.95	0.010	0.992	0.971	0.999
LayerCAM-Krum	0.517	0.517	0.069	0.879	0.516	0.724	0.337	0.337	0.095	0.834	0.337	0.621
LayerCAM-AE	**1.0**	**1.0**	**0.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**	**1.0**	**1.0**	**1.0**

to screen malicious model updates, which coincides with the precision of 0.516 for LayerCAM-Krum on non-IID SVHN, as shown in Table 4.

Compared to LayerCAM-AE, GradCAM-AE performs worse on the more complex non-IID CIFAR-100 dataset, as shown in Fig. 4b. The reason is that the local models trained by FL benign users on heterogeneous datasets cannot generate heat maps through GradCAM, resulting in GradCAM-AE's misclassification. To this end, the dataset can have an impact on GradCAM-AE. The more complex the dataset, the more difficult it is for GradCAM-AE to identify malicious users, as also confirmed in Figs. 2b and 4b. The remaining defense methods still undergo poor anomaly detection under the non-IID datasets, as they do on the IID datasets. The detection rates of ResNet-50 on SVHN and CIFAR-100 under the non-IID settings are given in Table 4.



(a) SVHN (b) CIFAR-100

Fig. 4. ResNet-50 on non-IID SVHN and CIFAR-100.

Fig. 5 demonstrate that LayerCAM-AE achieves approximately 70% test accuracy, markedly outperforming CosDefense, which only reaches around 1%. This performance gap stems from CosDefense's reliance on cosine similarity, which proves ineffective against advanced poisoning strategies such as those leveraging graph autoencoders. As a result, malicious users remain undetected and continue to influence the FL process.

We replace the ResNet-50 model with the REGNETY-800MF model. The trend of the test accuracy of the defense methods with the communication rounds is consistent with the observation in Fig. 4a, except for GradCAM-AE, as

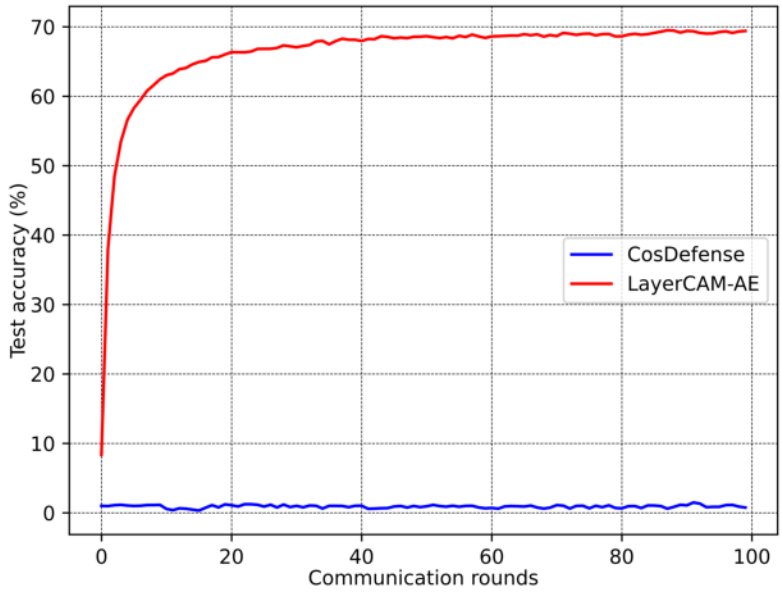


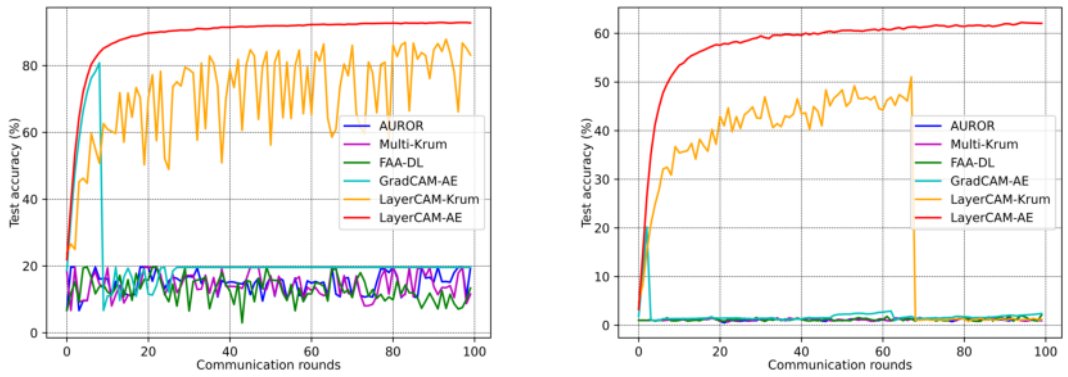
Fig. 5. The test accuracy of CosDefense and LayerCAM-AE on non-IID CIFAR-100.

Table 5. Detection rates of **REGNETY-800MF** on **non-IID** SVHN and CIFAR-100 with 3 attackers.

REGNETY-800MF	non-IID SVHN						non-IID CIFAR-100					
Methods	Recall	Precision	FPR	ACC	F1 score	AUC	Recall	Precision	FPR	ACC	F1 score	AUC
AUROR	0.057	0.042	0.153	0.748	0.047	0.452	0.013	0.008	0.158	0.738	0.01	0.427
Multi-Krum	0.073	0.073	0.132	0.768	0.073	0.470	0.1	0.1	0.129	0.775	0.1	0.486
FAA-DL	0.743	0.127	0.732	0.327	0.215	0.505	0.727	0.123	0.730	0.327	0.210	0.498
GradCAM-AE	0.848	-	0.004	0.977	0.844	0.922	0.828	0.828	0.02	0.974	0.820	0.917
LayerCAM-Krum	0.547	0.547	0.065	0.887	0.547	0.741	0.95	0.95	0.007	0.987	0.95	0.971
LayerCAM-AE	**1.0**	**1.0**	**0.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.0**	**1.0**	**1.0**	**1.0**

shown in Fig. 6a. The reason is that the performance of GradCAM may vary, depending on the architecture of the neural network analyzed. It may not be as effective for models with complex architectures, such as attention-based models, where the relationships between features are more intricate.

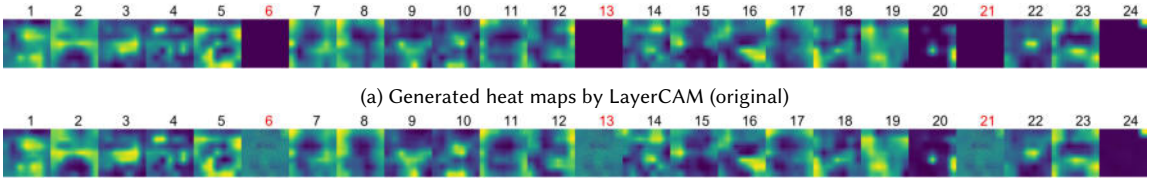
As contrasted with GradCAM-AE, LayerCAM-AE is designed to be compatible with various network architectures, including both traditional CNNs and more complex architectures such as ResNets or REGNETY-800MF. This adaptability allows LayerCAM-AE to be applied across a wide range of models without modification. In other words, LayerCAM-AE is unaffected by the structure of DNNs, as is also evident in Figs. 4a and 4b. In Fig. 6b, LayerCAM-Krum mistakenly selects the malicious local model update as the global model update in the 68th communication round, causing the test accuracy of the global model to drop sharply. The detection rates of REGNETY-800MF on SVHN and CIFAR-100 under the non-IID settings are given in Table 5.



(a) SVHN (b) CIFAR-100

Fig. 6. REGNETY-800MF on non-IID SVHN and CIFAR-100.

The considerable distinction is delineated in Table 6, where the server iteratively computes the average reconstruction errors across 24 users over 100 communication rounds. Fig. 7 showcases the original LayerCAM heat maps alongside the reconstructed heat maps generated by the autoencoder under REGNETY-800MF configuration in the non-IID CIFAR-100 setting with $\alpha = 0.1$. Fig. 7 (a) shows the heat maps generated by given 24 local model updates (from 21 benign users and 3 attackers), where No. 6, 13, and 21 are the ones based on malicious users' model updates. We can intuitively see that the heat maps generated by the attackers' model updates are significantly different from those of other benign users' model updates. Remarkably, the reconstruction errors in Table 6 for malicious users 6, 13, and 21 are all 0.214, which are significantly higher than those of the benign users, highlighting their anomalous behavior.



(a) Generated heat maps by LayerCAM (original)

(b) Reconstructed heat maps by autoencoder

Fig. 7. Comparison between generated heat maps by LayerCAM and reconstructed heat maps by the autoencoder under REGNETY-800MF on non-IID CIFAR-100 dataset with $\alpha = 0.1$, where No. 6, 13, and 21 are malicious users.

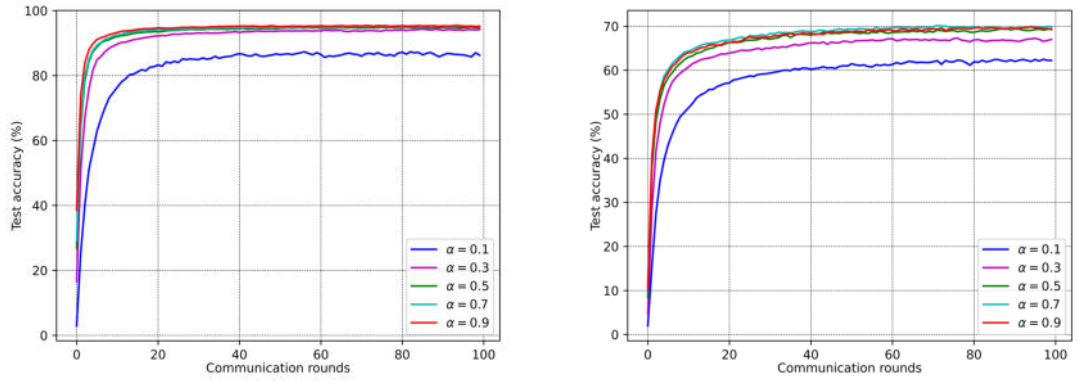
5.4 Scalability

Figs. 8 and 9 exhibit the global model's test accuracy of ResNet-50 and REGNETY-800MF under different degrees of heterogeneity ($\alpha = 0.1, 0.3, 0.5, 0.7$ and 0.9) on the SVHN and CIFAR-100 datasets, respectively. The experimental results indicate that LayerCAM-AE is nearly unaffected by the structures of DNNs and the degree of dataset heterogeneity.

To validate LayerCAM-AE on a larger dataset, we conduct experiments involving 21 users with 3 attackers over 100 communication rounds under a Non-IID Tiny-ImageNet-200 dataset setting. The detection rates for Recall, Precision,

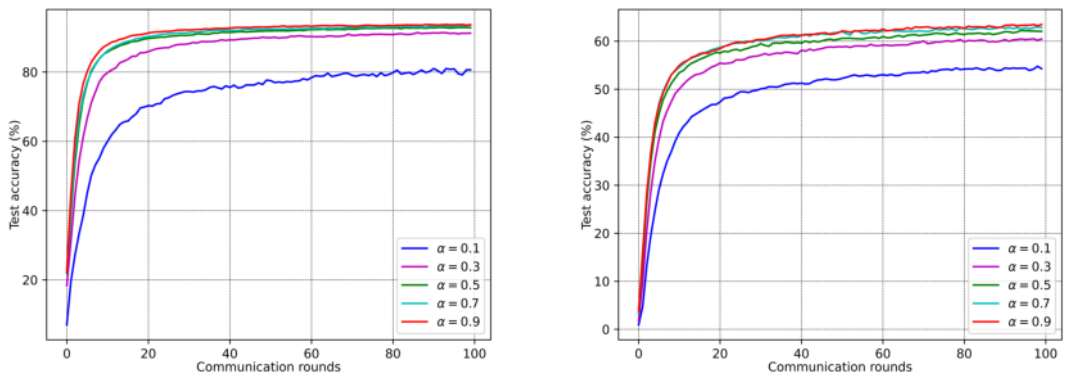
Table 6. Reconstruction errors for REGNETY-800MF on non-IID CIFAR-100 dataset with $\alpha = 0.1$

Index of users	reconstruction errors
1	0.007 ± 0.005
2	0.006 ± 0.004
3	0.007 ± 0.004
4	0.007 ± 0.006
5	0.007 ± 0.004
6	**0.214** ± 0.004
7	0.007 ± 0.004
8	0.007 ± 0.004
9	0.007 ± 0.005
10	0.007 ± 0.005
11	0.007 ± 0.004
12	0.007 ± 0.005
13	**0.214** ± 0.004
14	0.007 ± 0.004
15	0.006 ± 0.004
16	0.007 ± 0.005
17	0.007 ± 0.006
18	0.007 ± 0.004
18	0.007 ± 0.005
20	0.007 ± 0.005
21	**0.214** ± 0.004
22	0.006 ± 0.004
23	0.007 ± 0.005
24	0.007 ± 0.005



(a) SVHN (b) CIFAR-100

Fig. 8. The test accuracy of FL global model varies with the degree of heterogeneity (α) with regard to the ResNet-50 on SVHN and CIFAR-100 dataset.



(a) SVHN (b) CIFAR-100

Fig. 9. The test accuracy of FL global model varies with the degree of heterogeneity (α) with regard to the REGNETY-800MF on SVHN and CIFAR-100 dataset.

Accuracy, F1 score, and AUC in this challenging scenario are 0.928, 0.931, 0.988, 0.918, and 0.962, respectively, while the FPR increases to 0.238. This observed performance drop primarily stems from the highly non-IID nature of the dataset, where the features owned by local users may not contain labeled images sampled by the server. This discrepancy causes heat maps generated by benign users to differ significantly from those of other benign users, leading to them being occasionally misclassified as malicious. This issue typically arises in highly non-IID settings, especially when the dataset contains an exceptionally large number of classes. This perspective is further reinforced by the 12% decline in test accuracy observed in Fig. 10.

We extend our experiments to include a higher-resolution dataset, namely CelebA (128×128 pixels), under a non-IID partitioning scheme. The experimental results demonstrate that LayerCAM-AE maintains outstanding detection performance, achieving near-ideal performance across all evaluation metrics (Recall: 1.0, Precision: 1.0, FPR: 0.0, Accuracy: 1.0, F1 Score: 1.0, and AUC: 1.0). These findings confirm that the LayerCAM-AE framework generalizes effectively to higher-resolution datasets, preserving its discriminative power despite the more distributed nature of gradient information. Therefore, we believe that the distinctiveness of LayerCAM heat maps between benign and malicious models persists across different input resolutions, and the autoencoder-based representation learning mechanism remains highly sensitive to anomalous activation patterns, even in large-resolution and non-IID scenarios such as CelebA.

We conduct additional experiments to provide a visual comparison between the heat maps generated by benign and malicious users at varying resolutions. As shown in Fig. 11, we present LayerCAM-generated heat maps under the ResNet-50 model for the non-IID CIFAR-100 and non-IID CelebA datasets with $\alpha = 0.5$. Specifically, 24 local model updates are illustrated, 21 corresponding to benign users and 3 to malicious users (i.e., users No. 6, 13, and 21). The results demonstrate that the heat maps associated with malicious updates display distinct spatial activation patterns compared to those of benign users. In particular, the malicious models tend to generate displaced or irregular activation regions, indicative of abnormal feature extraction behavior induced by adversarial perturbations. Conversely, the benign

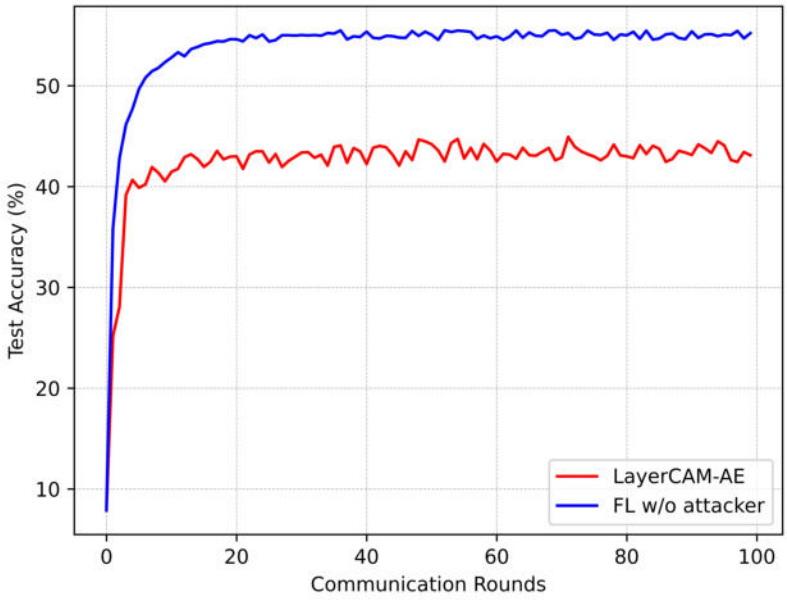
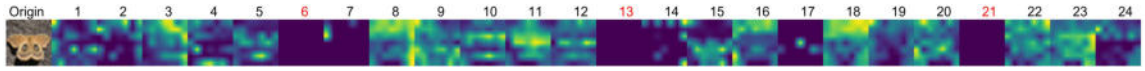


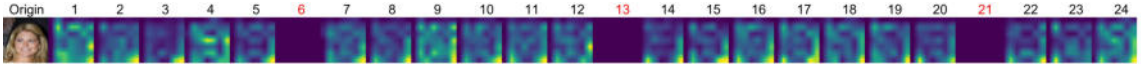
Fig. 10. The test accuracy of ResNet-50 on non-IID Tiny-Imagenet-200.

models consistently emphasize semantically meaningful areas of the input images, revealing stable and interpretable feature responses across users.

Furthermore, when comparing results across different resolutions, the CelebA dataset (with a higher input resolution of 128×128 pixels) yields more coherent and homogeneous heat maps among benign users compared to the lower-resolution CIFAR-100 dataset (32×32 pixels), as illustrated in Figs. 11a and 11b. This observation suggests that higher image resolution enhances the spatial consistency of benign users' activation representations, thereby improving the discriminability between benign and malicious heat maps. Notably, the distinction between benign and malicious activation patterns remains observable in both datasets, confirming that LayerCAM-AE effectively captures and differentiates anomalous activation behaviors across varying resolutions. These findings further validate the robustness of the proposed LayerCAM-AE framework to resolution variations in visual data.



(a) Generated heat maps under ResNet-50 on non-IID CIFAR-100 dataset (32×32 pixels) with $\alpha = 0.5$.



(b) Generated heat maps under ResNet-50 on non-IID CelebA dataset (128×128 pixels) with $\alpha = 0.5$.

Fig. 11. Comparison between generated heat maps by LayerCAM under ResNet-50 on non-IID CIFAR-100 dataset and non-IID CelebA dataset with $\alpha = 0.5$ in 6-th communication round.

To evaluate the scalability of LayerCAM-AE in larger FL systems, we increase the number of participating users from 24 to 100. Fig. 12 demonstrates that FL maintains a high test accuracy of 70% and 57.5% under the proposed LayerCAM-AE, when the number of users is set to 24 (21 benign users and 3 malicious users) and 100 (97 benign users and 3 malicious users), respectively. The decrease in test accuracy of global model is because given the total dataset size remains same, assigning the dataset to more users reduces the amount of data per user under non-IID (i.e., different users have data from different distributions) scenario, smaller local datasets exacerbate this heterogeneity, which causes the detection rates of LayerCAM-AE on 100 users in detecting malicious local models to decrease.

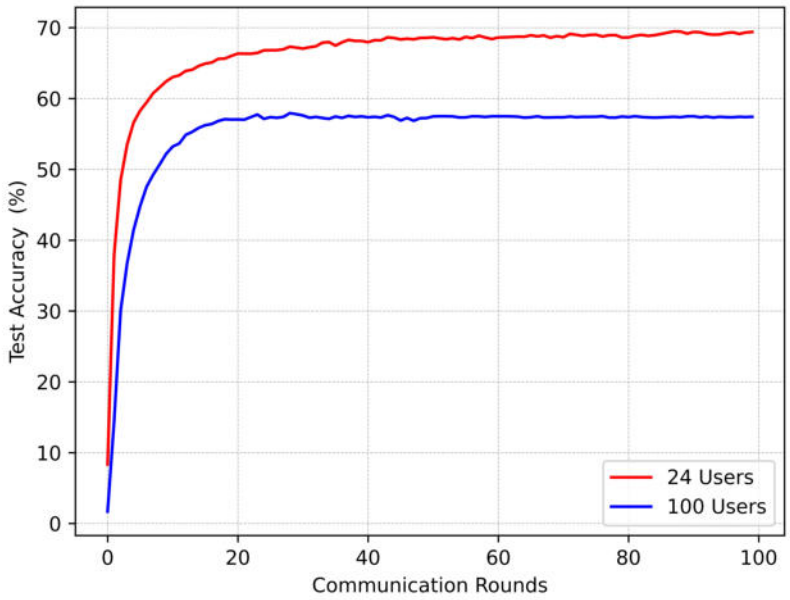


Fig. 12. The test accuracy of FL global model under the defense of LayerCAM-AE with 24 users vs. 100 users train ResNet-50 on non-IID CIFAR-100 dataset.

Moreover, to further evaluate the scalability of LayerCAM-AE, we define the attack rate as the ratio of the number of attackers to the total number of local users, i.e., ATR $= \frac{\#\text{attackers}}{\#\text{all users}} \times 100\%$. As depicted in Figure 13, LayerCAM-AE and LayerCAM-Krum are evaluated under non-IID conditions using the CIFAR-100 dataset with an imbalance factor ($\alpha = 0.5$), facing scenarios with 3 and 12 attackers. Given 24 local users, the ATR of LayerCAM-Krum from increases 12.5% to 50% as LayerCAM-Krum struggles to detect the attackers, the maximum gap of test accuracy is 49.950% (in the 67th communication round) much higher than 8.116% that is the maximum gap of test accuracy of LayerCAM-AE in the 96th communication round. This highlights the enhanced scalability and reliability of LayerCAM-AE.

Sensitivity analysis of thresholds. In the voting process, we use a rule that flags users as malicious if they are detected in ϵ out of every ξ communication rounds. A smaller ξ (e.g., 2) with a lower ϵ (e.g., 1) allows for quicker detection but may lead to false positives, especially in noisy or non-IID environments. Larger ξ (e.g., 5) with a higher ϵ (e.g., 3) offer more robustness but slower detection. To further explore the impact of the only tunable parameter, β,

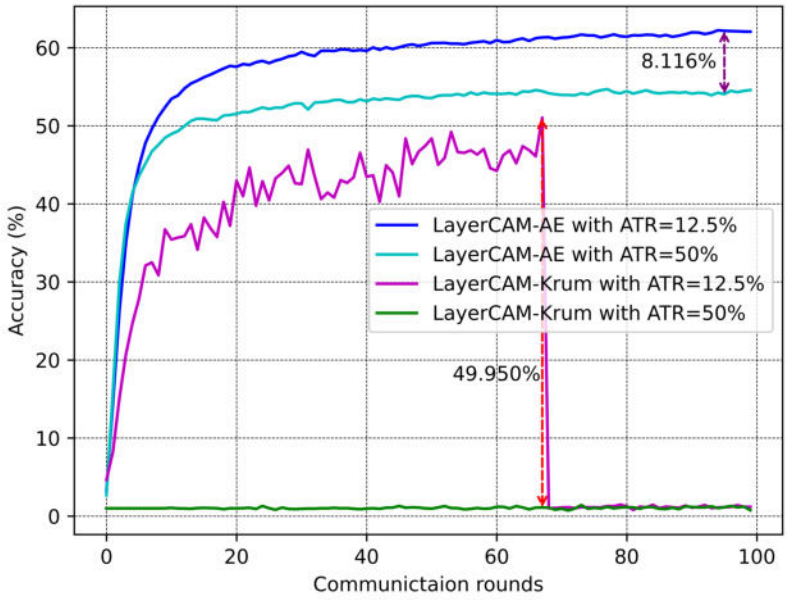


Fig. 13. Scalability test with respect to ATR.

associated with the threshold δ on our proposed LayerCAM-AE, we set the β to 0.5, 1.0, 1.5, 2.0 and 2.5 respectively. Table 7 summarizes the results of detection rates as the parameter β changes. When β is set to 0.5, FPR increases to 0.319. This is because a lower threshold results in more benign users being misclassified as malicious users. Accordingly, ACC, F1 score, and AUC are reduced to 0.721, 0.565, and 0.864, respectively. In contrast, when the β is set to 1.0, 1.5 and 2.0, the ACC, F1 score, and AUC always remain at 1.0. However, when the β is set to 2.5, Recall (also known as the true positive rate) drops to 0.833. This is because a larger threshold results in more malicious users being misclassified as benign users. Accordingly, ACC, F1 score, and AUC are also reduced to 0.979, 0.833, and 0.917, respectively. Therefore, a lower β (e.g., < 1.0) increases sensitivity and detects subtle anomalies early but risks higher false positives, while a higher β (e.g., > 2.0) reduces false alarms but may miss stealthy model poisoning attacks.

Table 7. Detection rates vary with parameter β.

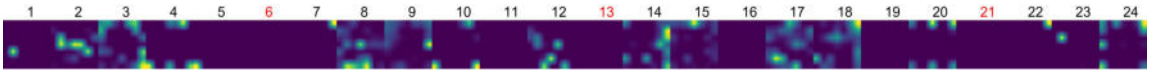
β	Recall	Precision	FPR	ACC	F1 score	AUC
0.5	1.0	0.438	0.319	0.721	0.565	0.864
1.0	1.0	1.0	0.0	1.0	1.0	1.0
1.5	1.0	1.0	0.0	1.0	1.0	1.0
2.0	1.0	1.0	0.0	1.0	1.0	1.0
2.5	0.833	1.0	0.0	0.979	0.833	0.917

In practice, ϵ and ξ should be tuned according to the application's risk tolerance and the degree of variability among model updates. For example, in high-risk domains like healthcare or finance, smaller ξ and lower ϵ might be preferred

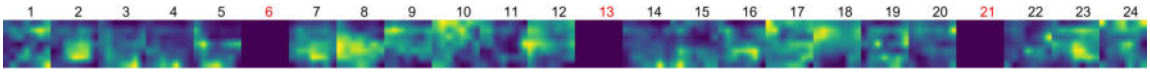
to ensure rapid detection, even at the cost of occasional false positives. Conversely, in environments with high user heterogeneity, such as mobile or IoT networks, more relaxed ϵ and larger ξ can help reduce false alarms caused by natural data drift. The only tunable parameter, β, associated with the threshold δ, requires adjustment based on the variability of model updates; values between 1.0 and 2.0 often often serve as a practical starting point.

5.5 Ablation Studies

Figs. 14a and 14b show the heat maps generated by GradCAM and LayerCAM on REGNETY-800MF on the non-IID ($\alpha = 0.1$) CIFAR-100 dataset in the sixth communication round, respectively. It is observed from the heat maps that LayerCAM is better at capturing image features than GradCAM. While users 5, 11, 16, and 19 are benign, the test images and their local model updates via the GradCAM process fail to generate heat maps and are mistakenly considered malicious. However, LayerCAM-AE does not produce misclassification because it accurately captures the image features. This can also be evidenced by the test accuracy of LayerCAM-AE in Figs. 4b, 6a, and 6b. To this end, LayerCAM is essential to the LayerCAM-AE architecture.



(a) heat maps generated by **GradCAM** of REGNETY-800MF with non-IID ($\alpha = 0.1$) CIFAR-100 dataset in 6-th communication round



(b) heat maps generated by **LayerCAM** of REGNETY-800MF with non-IID ($\alpha = 0.1$) CIFAR-100 dataset in 6-th communication round

Fig. 14. Comparison between generated heat maps by GradCAM and LayerCAM on REGNETY-800MF with non-IID ($\alpha = 0.1$) CIFAR-100 dataset in 6-th communication round, where No. 6, 13, and 21 are malicious users.

We replace the autoencoder in LayerCAM-AE with Krum, and plot the test accuracy of LayerCAM-Krum in Figs. 4a, 4b, 6a, and 6b. The test accuracy fluctuates significantly and fails to converge. There are even malicious local model updates involved in FL global model updates. This is because, in the absence of the autoencoder remapping of the LayerCAM heat maps, LayerCAM-Krum is unable to highlight the hidden features of the heat maps and improve the heat map distinguishability. For this reason, the autoencoder is indispensable for the LayerCAM-AE architecture.

5.6 Runtime

To measure the time cost, we use ResNet-50 to train FL with 21 benign users under the Non-IID ($\alpha = 0.5$) CIFAR-100 dataset to calculate the average runtime per communication round.

As shown in Fig. 15, the average runtime of our proposed LayerCAM-AE (61.02 s) is higher than that of Euclidean distance-based defense method, i.e., Multi-Krum (55.45 s). This is because the Euclidean distance-based defense does not require training, while LayerCAM-AE takes a little longer time to train the autoencoder. Nevertheless LayerCAM-AE is still faster than that of the machine learning-based AUROR (72.41 s) and FAA-DL (79.96 s). Moreover, it is also competitive with GradCAM-AE (61.36 s) and LayerCAM-Krum (59.02 s). It is noting that although runtime of LayerCAM-AE is not the fastest, detection rates of LayerCAM-AE are the highest.

Fig. 16 presents the average runtime per communication round for both LayerCAM (without an autoencoder) and LayerCAM-AE (with an autoencoder) as the number of participating users increases. As observed, the runtime grows with the number of users, reflecting higher computational demands in larger FL systems. The results show a gradual and

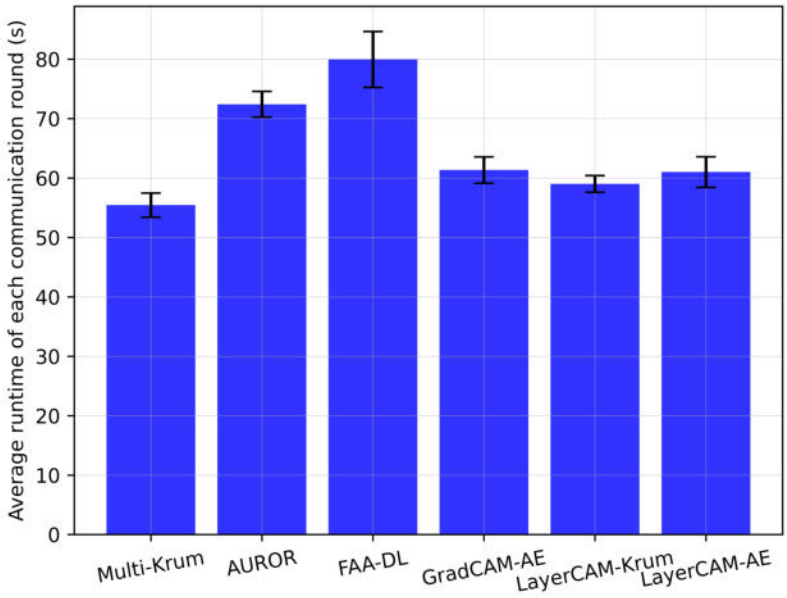


Fig. 15. Comparison of LayerCAM-AE's average runtime with other benchmarks.

linear increase in runtime as the user count rises, consistent with the expected scaling behavior in FL systems. While the integration of the autoencoder introduces a modest overhead for smaller user groups (e.g., approximately 1.647 s and 1.667 s for 24 and 48 users, compared to 0.847 s and 1.315 s for LayerCAM alone), this additional cost becomes negligible as the number of users expands. Beyond approximately 72 users, the runtime gap between LayerCAM-AE and LayerCAM narrows substantially, highlighting the scalability and computational efficiency of the proposed framework. This efficiency can be attributed to the parallelized processing of all LayerCAM-generated heat maps within the autoencoder, enabling concurrent computation on the server. Consequently, the per-round computational complexity of LayerCAM-AE scales approximately linearly with the number of users, similar to that of LayerCAM without the autoencoder. Therefore, LayerCAM-AE demonstrates excellent scalability and efficiency in large-scale FL systems, maintaining stable computational performance even as the system size increases.

5.7 Discussions of This Work

In this work, we explored leveraging enhanced LayerCAM visualization techniques to defend against poisoning attacks in FL. Although both integrated gradients [27] and LayerCAM are, in fact, visualization techniques, their methodologies and outputs are fundamentally distinct. Integrated gradients attributes a model's prediction to individual input features, like each pixel, by averaging gradients along a path from a baseline input to the actual input, resulting in fine-grained, pixel-level importance scores. In contrast, LayerCAM generates class-specific spatial heatmaps by combining convolutional feature maps with their gradients, highlighting which regions of an image contributed most to a prediction. Integrated gradients cannot directly substitute LayerCAM in this work because they serve different purposes,

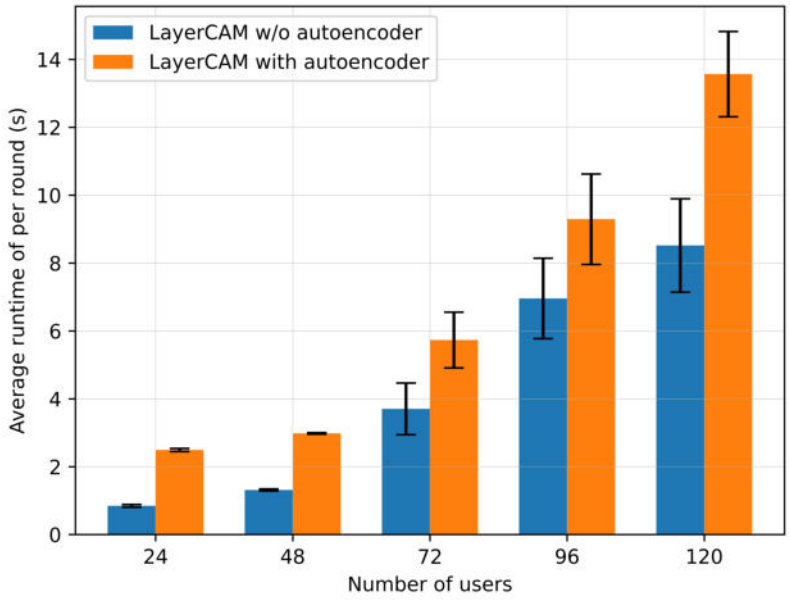


Fig. 16. The average runtime of LayerCAM without autoencoder and LayerCAM with autoencoder per round varies with the number of users.

produce incompatible output formats, and rely on different assumptions about model structure and interpretability. While integrated gradients are model-agnostic but incur higher computational overhead, LayerCAM is specifically designed to generate class-discriminative spatial heat maps in convolutional neural networks, aligning perfectly with the autoencoder-based anomaly detection approach used in our proposed LayerCAM-AE framework.

We acknowledge that parameter tuning can be challenging under certain extreme FL conditions. First, LayerCAM-AE assumes that the number of malicious users is smaller than the number of benign users. When malicious users constitute the majority, the aggregated heat maps may become dominated by abnormal activation patterns, potentially leading to degraded detection performance or even failure. This limitation arises because the collective representation used for model comparison is biased toward malicious contributions, reducing the distinguishability between benign and adversarial updates. Second, LayerCAM-AE's performance may be affected by highly non-IID data distributions, particularly in datasets containing a very large number of classes (e.g., ImageNet contains over 20,000 classes). In such cases, benign users whose local data do not share label overlap with the server's sampled classes may produce heat maps that differ substantially from those of other benign users. This inter-user inconsistency can occasionally cause benign users to be incorrectly identified as malicious.

Additionally, our current work primarily focuses on FL for image classification tasks. The broader applicability and effectiveness of the proposed LayerCAM-AE framework in other FL domains, such as natural language processing or time-series analysis, remain important directions for future research. Nevertheless, we believe that the core principles underlying LayerCAM-AE provide a strong foundation for anomaly detection across diverse data modalities. Specifically,

the mechanism of LayerCAM-AE, which analyzes feature map activations to identify anomalous model updates, is conceptually transferable to tasks where meaningful and interpretable intermediate representations can be extracted. Extending the framework to non-visual FL applications represents a promising and exciting avenue for our future work, potentially broadening the impact and generalizability of LayerCAM-AE beyond image-based settings.

We claim that LayerCAM-AE makes a significant contribution by providing a robust, purpose-built defense against model poisoning attacks, a highly critical and persistent threat unique to the FL update-sharing process. Nevertheless, we believe that integrating LayerCAM-AE with complementary defense techniques presents a promising direction for future work. For instance, combining our visualization-driven anomaly detection with other robust defense strategies could lead to a more holistic FL security framework capable of addressing a wider range of threat vectors. In our future research, we plan to extend the LayerCAM-AE architecture to include additional modules. These extensions will be designed to enhance its generalizability and robustness across a broader spectrum of advanced or hybrid attack scenarios that manipulate the federated learning process, thereby strengthening its utility beyond pure model poisoning.

6 Related Work

This section reviews the literature on defense models against poisoning attacks on FL.

Euclidean distance-based defense approaches. Several existing approaches, based on Euclidean distance, have been developed to detect poisoning attacks in FL scenarios. Within this framework, methods like Krum [1, 35] or Multi-Krum [1, 35] calculate a score for each local model update, which represents the sum of its Euclidean distance from neighboring updates. Multi-Krum identifies local model updates with high scores as potentially malicious and excludes them. In contrast, the Trimmed-mean method [32] is a coordinate-wise aggregation approach that aggregates each coordinate of the local model update independently. For each coordinate, the values of corresponding coordinates in the users' updates are sorted. The largest and smallest k values are then removed, and the trimmed mean calculates the average of the remaining values as the corresponding coordinate of the aggregated model update. To mitigate poisoning attacks involving numerous malicious users, FLDetector [34] has been developed to predict a user's model update in each communication round based on historical updates. If the received local model update from a user consistently differs from the predicted update across multiple communication rounds, it is flagged as malicious. Another defense strategy against poisoning attacks on FL [4] involves categorizing users into distinct groups and training a global model for each group using an existing FL aggregation rule. A majority vote mechanism, based on the global models of all groups, is then used to determine if a test input has been tampered with by an attacker.

Cosine-similarity based defense. Cosine similarity is used in [2] to measure the alignment of local model updates with the trusted global model update that is trained by clean dataset. The local model updates with low similarity are scaled down or ignored. [20] utilizes cosine similarity to measures the angular similarity between local model updates and a reference direction (e.g., the aggregated global model or the average of local model updates). The local model updates are clustered based on their angular similarities. The clusters with low cosine similarities to the majority of local model updates are flagged as suspicious and excluded from aggregation. Similarly, [40] keeps the underlying assumption is that the local gradients of benign user align closely with the direction of global gradient, whereas malicious gradients deviate significantly. The authors use cosine similarity to measure the angular alignment between the local gradients of users and the global gradients maintained by the server. The local gradients with obtuse angles (cosine similarity is less than zero) are flagged as malicious because they indicate opposing or disruptive directions. The server discards these local gradients to prevent them from affecting the global model's update.

Machine learning-based defense. To identify malicious local users while ensuring the generation of accurate models, a statistical mechanism known as AUROR has been introduced [25]. AUROR is founded on the observation that the primary model features from the majority of honest users demonstrate a consistent distribution, whereas those from malicious users exhibit an aberrant distribution. AUROR utilizes K-means to cluster uploaded local model updates across training communication rounds and eliminates malicious local model updates, i.e., contributions from small clusters that surpass a predefined distance threshold are flagged as malicious.

In another approach, the authors of [26] presented the Federated Anomaly Analytics enhanced Distributed Learning (FAA-DL) framework, which is a lightweight, unsupervised anomaly detection method based on support vector machine (SVM). FAA-DL employs an appropriate kernel function and soft margins to estimate a nonlinear decision boundary, effectively segregating benign and malicious local model updates. Given the criticality of detecting network attacks, the authors of [16] developed the FL framework, an integrated isolation forest algorithm, to identify and filter malicious local model updates before global model aggregation. They indicated that the leaf nodes representing the malicious model are closer to the root, facilitating their detection. Deep reinforcement learning methods were employed to dynamically adjust the detection threshold for identifying malicious local model updates. In combating generative adversarial network (GAN) attacks, the authors of [6] devised a system that isolates local model update parameters from all users, preventing attackers from setting up GANs to carry out attacks.

The current defense mechanisms against malicious local model updates, particularly those based on Euclidean distances, face significant challenges related to the “curse of dimensionality" in the context of DNNs. This is particularly relevant considering that local model update parameters can encompass millions or even billions of parameters, including both weights and biases. Within high-dimensional spaces, Euclidean distances may inadvertently exaggerate the distances between model updates, leading to reduced effectiveness in detecting malicious updates. Machine learning-based detection mechanisms often heavily rely on fine-tuning hyperparameters and setting thresholds with precision. However, recent experimental findings [8, 36] suggest that this approach may be prone to ineffective anomaly detection.

On the contrary, the proposed LayerCAM-AE framework introduces a novel approach to detecting malicious local model updates in FL by leveraging LayerCAM, thus departing from conventional Euclidean distance metrics. The LayerCAM-AE defense mechanism against poisoning attacks on FL is adept at transforming the high-dimensional local model parameters—often numbering in the millions—into low-dimensional and fine-grained LayerCAM heat maps. Moreover, an is deployed alongside LayerCAM to improve the distinguishability of the heat maps, thus boosting the accuracy in identifying abnormal heat maps generated by malicious local model updates.

7 Conclusion

In this paper, we proposed LayerCAM-AE, a new LayerCAM and autoencoder-assisted defense mechanism against model poisoning attacks on FL. The defense mechanism was designed to process the local model updates by employing a specific image from the test dataset to generate visual LayerCAM heat maps. The autoencoder was extended to enhance the visibility of hidden features within the LayerCAM heat maps. Moreover, a voting algorithm was developed to consistently filter out transient malicious model updates, thereby reducing the likelihood of erroneously identifying malicious local models. Experimental results showed that the proposed LayerCAM-AE offers superior detection rates (Recall: 1.0, Precision: 1.0, FPR: 0.0, Accuracy: 1.0, F1 score: 1.0, AUC: 1.0) and FL test accuracy (69%) with RestNet-50 on non-IID CIFAR-100 dataset and significantly surpasses contemporary defense approaches under various settings.

Acknowledgments

This work was supported by national funds through FCT/MECI (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDP/UIDB/04234/2020) and under the project Intelligent Systems Associate Laboratory - LASI (LA/P/0104/2020) and project ADANET (PTDC/EEICOM/3362/2021).

References

[1] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017), pp. 118–128.

[2] Cao, X., Fang, M., Liu, J., and Gong, N. Z. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021* (2021), The Internet Society.

[3] Cao, X., and Gong, N. Z. MPAF: Model poisoning attacks to federated learning based on fake clients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (June 2022), pp. 3396–3404.

[4] Cao, X., Zhang, Z., Jia, J., and Gong, N. Z. Flcert: Provably secure federated learning against poisoning attacks. *IEEE Transactions on Information Forensics and Security 17* (2022), 3691–3705.

[5] Chattopadhay, A., Sarkar, A., Howlader, P., and Balasubramanian, V. N. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2018), pp. 839–847.

[6] Chen, Z., Fu, A., Zhang, Y., Liu, Z., Zeng, F., and Deng, R. H. Secure collaborative deep learning against gan attacks in the internet of things. *IEEE Internet of Things Journal 8*, 7 (2021), 5839–5849.

[7] Fang, M., Cao, X., Jia, J., and Gong, N. Z. Local model poisoning attacks to byzantine-robust federated learning. In *Proceedings of the 29th USENIX Conference on Security Symposium* (2020), pp. 1623–1640.

[8] Han, S., Hu, X., Huang, H., Jiang, M., and Zhao, Y. Adbench: Anomaly detection benchmark. In *Advances in Neural Information Processing Systems* (2022), S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., pp. 32142–32159.

[9] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[10] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., and Adam, H. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (October 2019).

[11] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 4700–4708.

[12] Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., and Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR abs/1602.07360* (2016).

[13] Jiang, P.-T., Zhang, C.-B., Hou, Q., Cheng, M.-M., and Wei, Y. Layercam: Exploring hierarchical class activation maps for localization. *IEEE Transactions on Image Processing 30* (2021), 5875–5888.

[14] Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. (2009), Toronto, ON, Canada.

[15] Li, K., Zheng, J., Yuan, X., Ni, W., Akan, O. B., and Poor, H. V. Data-agnostic model poisoning against federated learning: A graph autoencoder approach. *IEEE Transactions on Information Forensics and Security* (2024).

[16] Liu, W., Lin, H., Wang, X., Hu, J., Kaddoum, G., Piran, M. J., and Alamri, A. D2mif: A malicious model detection mechanism for federated-learning-empowered artificial intelligence of things. *IEEE Internet of Things Journal 10*, 3 (2023), 2141–2151.

[17] Liu, Z., Luo, P., Wang, X., and Tang, X. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August 15*, 2018 (2018), 11.

[18] McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (20–22 Apr 2017), A. Singh and J. Zhu, Eds., vol. 54 of *Proceedings of Machine Learning Research*, PMLR, pp. 1273–1282.

[19] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y., et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning* (2011), vol. 2011, Granada, Spain, p. 7.

[20] Nguyen, T. D., Rieger, P., De Viti, R., Chen, H., Brandenburg, B. B., Yalame, H., Möllering, H., Fereidooni, H., Marchal, S., Miettinen, M., et al. FLAME: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 22)* (2022), pp. 1415–1432.

[21] Pillutla, K., Kakade, S. M., and Harchaoui, Z. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing 70* (2022), 1142–1154.

[22] Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 10428–10436.

[23] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 618–626.

[24] Shejwalkar, V., and Houmansadr, A. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS* (2021).

[25] Shen, S., Tople, S., and Saxena, P. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications* (2016), pp. 508–519.

[26] Shi, S., Hu, C., Wang, D., Zhu, Y., and Han, Z. Federated anomaly analytics for local model poisoning attack. *IEEE Journal on Selected Areas in Communications 40*, 2 (2022), 596–610.

[27] Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (2017), D. Precup and Y. W. Teh, Eds., vol. 70 of *Proceedings of Machine Learning Research*, PMLR, pp. 3319–3328.

[28] Wang, J., Guo, S., Xie, X., and Qi, H. Federated unlearning via class-discriminative pruning. In *Proceedings of the ACM Web Conference 2022* (New York, NY, USA, 2022), WWW '22, Association for Computing Machinery, p. 622–632.

[29] Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., and Chan, K. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications 37*, 6 (2019), 1205–1221.

[30] Yaldiz, D. N., Zhang, T., and Avestimehr, S. Secure federated learning against model poisoning attacks via client filtering. *ICLR abs/2304.00160* (2023).

[31] Yao, L., and Miller, J. Tiny imagenet classification with convolutional neural networks. *CS 231N 2*, 5 (2015), 8.

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

[33] Zhang, Y., Bai, G., Chamikara, M. A. P., Ma, M., Shen, L., Wang, J., Nepal, S., Xue, M., Wang, L., and Liu, J. Agrevader: Poisoning membership inference against byzantine-robust federated learning. In *Proceedings of the ACM Web Conference 2023* (New York, NY, USA, 2023), WWW '23, Association for Computing Machinery, p. 2371–2382.

[34] Zhang, Z., Cao, X., Jia, J., and Gong, N. Z. Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2022), pp. 2545–2555.

[35] Zhang, Z., Su, Q., and Sun, X. Dim-krum: Backdoor-resistant federated learning for NLP with dimension-wise krum-based aggregation. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022* (2022), Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds., Association for Computational Linguistics, pp. 339–354.

[36] Zhao, Y., Rossi, R., and Akoglu, L. Automatic unsupervised outlier model selection. In *Advances in Neural Information Processing Systems* (2021), M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., pp. 4489–4502.

[37] Zheng, J., Li, K., Yuan, X., Ni, W., Tovar, E., and Crowcroft, J. Exploring visual explanations for defending federated learning against poisoning attacks. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking* (New York, NY, USA, 2024), ACM MobiCom '24, Association for Computing Machinery, p. 1596–1598.

[38] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016).

[39] Zhou, H., Yang, G., Dai, H., and Liu, G. PFLF: Privacy-preserving federated learning framework for edge computing. *IEEE Transactions on Information Forensics and Security 17* (2022), 1905–1918.

[40] Zhu, T., Guo, Z., Yao, C., Tan, J., Dou, S., Wang, W., and Han, Z. Byzantine-robust federated learning via cosine similarity aggregation. *Computer Networks 254* (2024), 110730.