Supplementary material of FLAD

1. Contributions of Our FLAD 1
2. Proximity Analysis of Tolerance for Malicious Attacks..... 1
3. Experimental Results on Adaptive Local Model Attack..... 2
4. Parameters Setting of Attacks 5
5. Visualizations of DBSCAN 6
6. Experimental Results on Non-IID Data..... 7
7. Ablation Experiments 8

1. Contributions of Our FLAD

Our main contributions are summarized below.

- We propose a Byzantine-robust federation learning scheme based on the idea of anomaly detection: FLAD, which builds an anomaly detector based on neural networks and clustering models to detect and reject anomalous parameters uploaded by malicious clients, truly achieving aggregation only for honest parameters.
- FLAD is the first model attack defense scheme that uses neural network models to adaptively learn the internal features of honest parameters and thus measure the degree of similarity of different parameters. The scheme avoids the limitations that exist in traditional measures of similarity using cosine similarity, Pearson correlation coefficient, L2 distance, etc.
- In FLAD, clean server datasets are used to bootstrap trust in local parameters. Thus, the number of malicious attackers can change dynamically in each iteration and the method is fully applicable when there are a large number or even up to half of the attackers in the system.
- We conducted extensive experiments on real datasets to evaluate our FLAD. the effectiveness of FLAD was comprehensively evaluated by using different datasets, various different attack and defense methods, different attacker number fractions, etc. The experimental results show that FLAD has high global accuracy and good robustness against various existing poisoning attacks, even including adaptive attacks.

2. Proximity Analysis of Tolerance for Malicious Attacks

This section is used to analyze the critical point of Byzantine attacks tolerated by FLAD.

The training objective of the server feature extraction model is to obtain optimal neural network parameters w_i^* and b_i^*, $i=1,\cdots,k$, where the loss function J uses MSE:

$$\operatorname{argmin}_{w_i^*,b_i^*}\sum_{w\in W_s} J\left(\sigma\left(f_k\circ f_{k-1}\circ\ldots\circ f_2\circ f_1(w)\right)-1\right).$$

Finally, we can obtain the extracted feature vectors for num server model parameters.

$$f_{c_i} = \sigma(w_{c_i} \times \prod_{j=1}^{k} w_j^* + b_1^* \times \prod_{j=2}^{k} w_j^* + \cdots + b_k^*).$$

Therefore, the key parameter of the DBSCAN algorithm, the scan radius

$$eps = ||\max f_{c_i} - \min f_{c_i}||,$$

where $||\cdot||$ denotes the L2 distance. The central value used to measure the standard of honest parameters:

$$\mu = \frac{1}{num} \sum_{i=1}^{num} f_{c_i}.$$

The local parameters w_i uploaded by the clients are fed into the feature extraction model to obtain the corresponding feature extraction vectors $F = \{f_1, \cdots, f_i, \cdots, f_n\}$:

$$f_i = \sigma(w_i \times \prod_{j=1}^{k} w_j^* + b_1^* \times \prod_{j=2}^{k} w_j^* + \cdots + b_k^*).$$

Assumption 1. There are no less than 3 honest parameters in the local parameter w_i, and the client feature vector with the closest L2 distance from the central value μ is f_q:

$$f_q = \text{argmin}_{f_i \in F} ||f_i - \mu||,$$

so f_q is considered as the feature extraction vector of the honest client q and w_q is the honest local parameter.

Since there is an honest parameter in w_i and the feature f_q corresponding to w_q is closest to the features of the honest parameter, it is reasonable to assume that w_q uploaded by client q is the honest parameter.

Definition 1. There exists a set of vectors $F = \{f_1, f_2, \cdots\}$. If there exists no less than three elements $\{f_y, \cdots\}$ in F within L2 distance eps of the vector f_x, f_y is said to be $1eps - f_x$ directly connected point. If there are also no less than three elements $\{f_z, \cdots\}$ in F within the L2 distance eps of the vector f_y, f_z is said to be $2eps - f_x$ directly connected point, and so on.

The set of feature extraction vectors for each client is $\{f_1, f_2, \cdots f_q, \cdots, f_n\}$, and let there exist at most $keps - f_q$ directly connected point. According to the principle of DBSCAN algorithm, the feature vector f_i with the farthest L2 distance from f_q and reachable density with f_q has $||f_i - f_q|| \leq keps$. Thus, the criticality of the honest and malicious parameters has the following theorem 1.

Theorem 1. A sufficient condition for the malicious parameter feature f_m to be successfully detected is $||f_m - f_q|| \geq (k+1)eps \geq |||f_i - f_q|| + eps$. A necessary condition for the honest parameter feature f_h to be successfully identified is $||f_n - f_q|| \leq keps$.

3. Experimental Results on Adaptive Local Model Attack

Adaptive Local Model Attack

In the context of the full-knowledge attack, attacker knows our Byzantine-robust defense scheme in advance. Attacker can use the state-of-the-art adaptive attack framework in research (Fang et al., 2020) to optimize the local model poisoning attack against our scheme. This adaptive attack treats the attack as an optimization problem in which the attackers dynamically optimize the malicious local model parameters at each iteration, such that the global model gradient after aggregation changes maximally in the opposite direction as it would have done in the absence of the attack. The general form of this attack optimization problem is

$$\text{subject to} \quad W = \mathcal{A}(w_1, \cdots, w_c, w_{c+1}, \cdots, w_n), \tag{3-1}$$

where the first c clients are assumed to be malicious. w_i' is the poisoned local model parameter uploaded by the ith malicious client. $\mathbf{s}$ is the column vector of the direction of change of the global model parameters in the absence of attack.

Next, we will generate strong Adaptive Local Model Poisoning Attack against our Byzantine-robust defense scheme FLAD by using the above framework.

Without loss of generality, we assume that $1 \sim (n-c)$ is the honest clients and $(n-c+1) \sim n$ is the malicious clients among n clients. In the general case, FLAD aggregates at each iteration to obtain the global model as follows:

$$W = \mathcal{A}(w_1, w_2, \cdots, w_{n-c}), \tag{3-2}$$

where $\mathcal{A}$ is the FedAvg. Therefore, the attacker's goal is to carefully construct the malicious parameters to confuse the feature extraction model in the anomaly detector so that the model outputs the wrong feature extraction vector set for the malicious parameters, thus misleading the malicious client detection algorithm to classify them as honest parameters for global aggregation. In the case of this attack, the poisoned global model obtained by aggregation is as follows:

$$W' = \mathcal{A}(w_1, w_2, \cdots, w_{n-c}, w_{n-c+1}', \cdots, w_n') \tag{3-3}$$

Bringing formulas (3-2), (3-3) into the adaptive attack framework (3-1), the adaptive model attack can be considered as the following optimization problem:

$$\max_{w_{n-c+1}', \cdots, w_n'} \mathbf{s}^T (W - W'). \tag{3-4}$$

We assume that the number of local training dataset for each client is equal. $\mathbf{s}^T(W - W')$ can be defined as:

$$\begin{aligned} \mathbf{s}^T (W - W') = & \quad -\mathbf{s}^T \frac{1}{n} (w_{n-c+1}' + \cdots w_n') \\ & + \mathbf{s}^T \left(\frac{1}{n-c} - \frac{1}{n}\right)(w_1 + \cdots w_{n-c}). \end{aligned} \tag{3-5}$$

Since the values of the honest parameters w_i do not affect the results of the optimization problem at each iteration round. Therefore, the optimization problem (3-4) can be simplified as:

$$\min_{w_{n-c+1}', \cdots, w_n'} \mathbf{s}^T (w_{n-c+1}' + \cdots + w_n'). \tag{3-6}$$

The goal of the attacker is to confuse the feature extraction model and mislead the malicious client detection algorithm while constructing the malicious parameters w' in the opposite direction of the global update in the absence of attack as much as possible. The solution to the optimization problem translates into generating holistic confused samples of the feature extraction model based on deviating the original honest parameters as much as possible.

Here we borrow the idea of FGSM (Fast Gradient Sign Method) to generate adversarial samples to solve the optimization problem (3-6) and obtain the adaptive attack of FLAD as shown in Algorithm 1. We consider all malicious client parameters as a whole and jointly solve the updated gradient with respect to the feature extraction model. Specifically, we take the honest parameter w_i as the initial value of the malicious parameter w_j' and add random noise u to w_j', thus avoiding the problem that the initial w_j' has a loss function of 0 on the feature extraction model, where u obeys a multivariate Gaussian distribution $\mathcal{N}(0, \sigma^2 \mathbf{I})$.

Algorithm 1 Adaptive Local Model Attack on FLAD

Algorithm Adaptive Local Model Attack on FLAD

Input: w_i: Local parameters sent by honest clients, $i \in \{1, 2 \cdots, n-c\}$;
F_{fc}: Feature extraction model;
ϵ: Step size of optimization per iteration;

Output: w'_j: Malicious parameters carefully constructed by the malicious clients, $j \in \{n-c+1, \cdots, n\}$;

$Pro_i \leftarrow LocalParameterFeature(F_{fc}, w_i), i \in \{1, 2 \cdots, n-c\}$;
Initialize $w'_j, Pro'_j \leftarrow w_i, Pro_i$;
$w'_j \leftarrow w'_j + \epsilon \times u, u \sim \mathcal{N}\left(0, \sigma^2 \mathbf{I}\right)$;
while True **do**
  $B'_j \leftarrow w'_j$;
  $Pro'_j \leftarrow LocalParameterFeature(F_{fc}, w'_j)$;
  $loss \leftarrow MSE(Pro'_j, Pro_i)$;
  Update $w'_j \leftarrow w'_j + \epsilon \times \text{sgn}(\nabla loss(w'_j))$;
  $\boldsymbol{w} = \left\{w_i, w'_j\right\}, i \in \{1, \cdots, n-c\}, j \in \{n-c+1, \cdots, n\}$;
  if $F_{fc}(\boldsymbol{w})$ is able to detect malicious clients **then**
    Break;
  end if
end while
return $w'_j \leftarrow B'_j$;

Algorithm *Local Parameter Feature Extraction* is used to obtain the feature extraction vector set Pro'_j corresponding to the current w'_j, and takes the difference between Pro'_j and Pro_i with honest parameters as the loss to solve for the updated gradient $\nabla loss\left(w'_j\right)$ of w'_j in the feature extraction model. Following the FGSM idea, the new optimized w'_j is obtained with ϵ as the step size (set to 0.01 in the experiments). When w'_j deviates too much to be successfully detected by the Algorithm *Malicious Client Detection*, the optimization result B'_j from the previous iteration is returned as the optimal adaptive malicious parameter w'_j.

Experimental Results

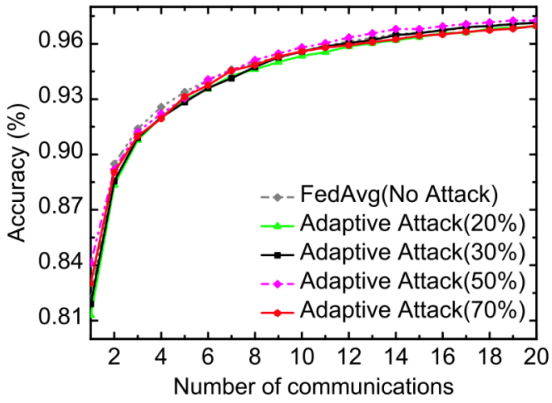


Figure 1. Global accuracy of MNIST versus the number of interactions for the adaptive attack with different percentage of malicious clients

We implement the adaptive local model attack on FLAD. The Figure 1 shows the model

accuracy versus the number of interactions for the adaptive attack with different percentage of malicious clients. We observe that even with malicious client proportions as high as 70%, the convergence rate and final accuracy of our scheme under adaptive attack are similar to FedAvg without attack, and the final global accuracy can be as high as 0.97. Therefore, the current state-of-the-art adaptive local model attack is not effective against our FLAD. The results show that when the attack strength is weak enough for FLAD to misjudge the malicious parameters, the malicious parameters are close enough to the honest parameters to have almost no effect on the accuracy of the aggregated global model.

Due to time constraints, we only carried out adaptive attacks on the MNIST dataset in the IID scenario. We will add more experimental results for the CIFAR-10 dataset and the non-IID scenario in revised manuscript.

4. Parameters Setting of Attacks

In our experiments, we consider both data poisoning attack and model poisoning attack. For model poisoning attack, we choose Gaussian Attack, Sign-flipping Attack and Zero-gradient Attack. For data poisoning attack, we choose the currently popular Label-flipping Attack and Backdoor Poisoning Attack. Furthermore, we assume that these attacks are implemented in the context of full knowledge to show that our FLAD can resist powerful poisoning attacks.

(1) Gaussian Attack

The malicious client sends parameters that obey Gaussian distribution similar to the honest parameters to the server, i.e., the malicious attacker will choose random variables from a Gaussian distribution with mean $\frac{1}{n-c}\sum_{i=1}^{n-c} w_i^t$ and variance 30 as his own local parameters w_j^t.

(2) Sign-flipping Attack

The malicious client makes the aggregated global model update in the opposite direction from when there is no attack by constructing malicious parameters. In our experiments, the attacker j sends $w_j^t = \mu \cdot \frac{1}{n-c}\sum_{i=1}^{n-c} w_i^t$ to the server, where the magnitude μ is equal to -3.

(3) Zero-gradient Attack

For the FedAvg, when the malicious parameters uploaded by the attackers and the honest parameters cancel each other after weighted summation, it creates the Zero-gradient problem where the global model is updated to 0 after server aggregation. Based on this, the malicious clients can upload $w_j^t = -\frac{1}{c}\sum_{i=1}^{n-c} w_i^t$ to the server, thus causing the Zero-gradient Attack on the global model.

(4) Label-flipping Attack

Label-flipping Attack flips the labels of the training samples to the target labels, but keeps the original features of the sample data unchanged. In our experiments, we relabel the source label of the malicious client local dataset as the target label after adding 1. For the MNIST dataset, if the source label is 9, the target label becomes 0. For the CIFAR-10 dataset, if the source label is truck, the target label becomes airplane. The malicious attacker trains the local model as an honest client after relabeling the local training dataset.

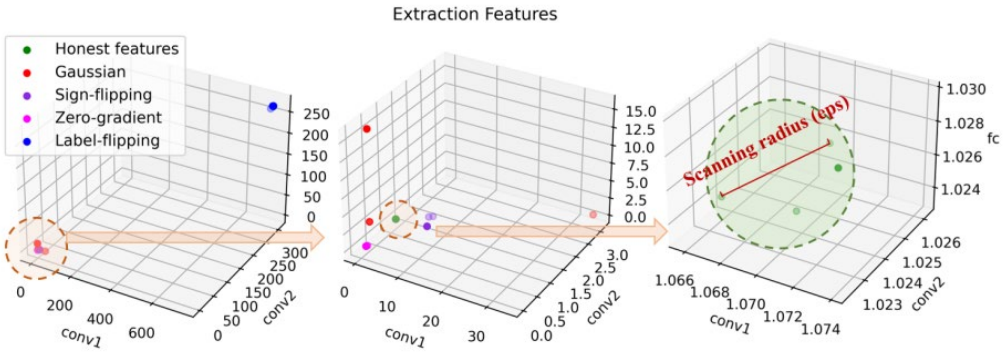
(5) Backdoor Poisoning Attack

Backdoor Poisoning Attack requires the attacker to implant the backdoor trigger in the source training data so that the final federated learning model will function properly on clean data, but will constantly predict the target class chosen by the attacker when data containing the trigger is present. In our experiments, we add 6×6 square pixel blocks in the top left corner of the malicious client local dataset as backdoor trigger. For the MNIST dataset, we change the target label of the image with the 6×6 black pixel block added to 0. For the CIFAR-10 dataset, we add 6×6 pixel blocks corresponding to the channel color on each of the three channels and change the target label of the three channel images with the backdoor added to airplane. After adding the backdoor to the local dataset and changing the target label, the malicious client starts to train the local model as an honest client.

Moreover, for Label-flipping Attack and Backdoor Poisoning Attack, following the idea of researches (Bagdasaryan et al., 2020; Sun et al., 2019), in order to mitigate the dilution of the attack effect due to global aggregation, we extend the poisoning local parameters by setting the weight factor $\gamma = n$ and send the extended local parameters to the server , where n is the total number of clients.

5. Visualizations of DBSCAN

After the local parameters of each client are input into the feature extraction model, the corresponding feature vector set can be obtained. Figure 2 shows a diagram of DBSCAN being used to partition client parameters. In order to illustrate the difference between honest and malicious parameters, Figure 2 also contains feature extraction vectors of malicious parameters under four different attacks. It can be seen from Figure 2 that the eigenvalues extracted by the honest parameter are highly similar and concentrated in the circle shown in Figure 2(c), and their scanning radius eps is on the order of 10^{-3}.



(a) (b) (c)

Figure 2. DBSCAN algorithm is used for the division of honest and malicious parameter features. (The dots represent feature extraction vectors of three local parameters of the client in the first iteration of the CNN model under the MNIST dataset)

Regarding the eps parameter. As shown in Figure 2(c), the setting of key parameter scanning radius eps in DBSCAN is related to the output of server model parameters in feature extraction model. Specifically, the server model parameter $W_s = \{w_{C_1}; w_{C_2}; \cdots; w_{C_{num}}\}$ input to feature extraction model F_{fc} to get output $\boldsymbol{p}_s = \{p_{C_1}; p_{C_2}; \cdots; p_{C_{num}}\}$, then $eps = ||\max \boldsymbol{p}_s - \min \boldsymbol{p}_s||$, where $||\cdot||$ is L2 norm. That is, the scanning radius of DBSCAN is determined by the server data set to avoid many false positives caused by artificial parameter setting.

6. Experimental Results on Non-IID Data

Non-IID data setup.

Similar to the previous study (Cao et al., 2021), we define the distribution probability q to measure the distribution difference between different client data. Suppose the dataset has a total of M categories. Randomly divide clients into M groups. If the probability that a training sample labeled l is allocated to group l is $q > 0$, the probability that the sample is allocated to another group is $(1-q)/(M-1)$. Within the same group, the data is evenly distributed to each client. The larger the value of q, the higher the degree of Non-IID between the client's local training data. In particular, if $q = 1/M$, the local training data of the client is IID. Therefore, we set $q > 1/M$ by default to simulate a scenario that is Non-IID.

Here we use the MNIST dataset as an example to demonstrate the data distribution process. Suppose we have a total of 50 clients and set $q = 0.4$. For MNIST, $M = 10$. We start by randomly dividing the clients into 10 groups, each containing 5 clients. For the training image with the number l (e.g., $l = 5$), we first assign it to Group 5 with a probability of 0.4, then to any other group with a probability of $(1-0.4)/(10-1) \approx 0.067$. Once the group is identified, for example, Group 5 is selected, we will randomly select a client from Group 5 and assign this training image to the selected client.

Experimental Results.

We set $q = 0.8$ on the MNIST to obtain the global model accuracy under different attack and defense methods, and the results are shown in Table 1. The results show that our scheme is still effective on Non-IID data.

Table 1. Global accuracy under different attack and defense methods (attack ratio=0.2, q=0.8)

	FedAvg	FLAD
NO ATTACK	0.949	**0.952**
GAUSSIAN	0.091	**0.949**
SIGN-FLIPPING	0.314	**0.950**
ZERO-GRADIENT	0.100	**0.947**
LABEL-FLIPPING	0.098	**0.952**
BACKDOOR	0.098	**0.951**

Due to time constraints, we have only verified the effectiveness of our scheme on Non-IID data at present. In the revised manuscript, we will also increase the effect comparison with the mainstream defense scheme on Non-IID data.

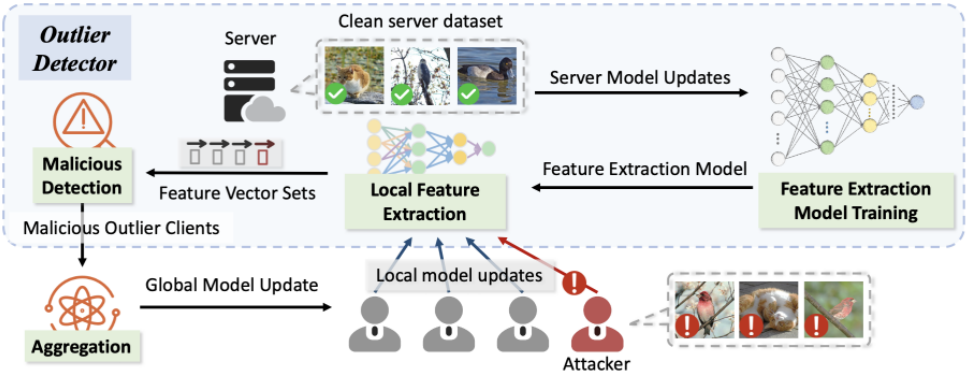


Figure 3. Overview of FLAD

7. Ablation Experiments

As shown in Figure 3, the FLAD has three key components: the server dataset, the feature extraction model of the fully connected network, and the DBSCAN clustering for malicious detection. According to the usage characteristics of each component, three variants of FLAD were considered for ablation experiments.

- **FLAD-withServer:** Aggregate server model parameters with honest client parameters after removing malicious nodes (That is, on the basis of FLAD, add the model of server data set for aggregation).
- **FLAD-NoDBSCAN:** Let the mean value of feature extraction value $\boldsymbol{p}_s = \{p_{C_1}; p_{C_2}; \cdots; p_{C_{num}}\}$ of server model parameters be center μ, $eps = ||\max \boldsymbol{p}_s - \min \boldsymbol{p}_s||$. If the output range of the feature extraction model of each client parameter is $[\mu - eps, \mu + eps]$, it is honest parameter, otherwise it is malicious parameter.
- **FLAD-NoFCNN:** The server does not use the feature extraction model and directly adopts DBSCAN algorithm to cluster. The scan radius parameter eps is the range of L2 distance between two server model parameters.

Table 2. The results of the ablation experiment (the global accuracy/the correct division rate)

	NO ATTACK	GAUSSIAN	SIGN-FLIPPING	ZERO-GRADIENT	LABEL-FLIPPING	BACKDOOR
FLAD-withServer	0.940/ 100%	0.944/ 100%	0.945/ 100%	0.946/ 99.8%	0.948/ 100%	0.941/ 100%
FLAD-NoDBSCAN	0.939/ 89.9%	0.938/ 87%	0.320/ 9.2%	0.932/ 94%	0.941/ 87%	0.928/ 58%
FLAD-NoFCNN	0.456/ 8.67%	0.383/ 30%	0.561/ 29.2%	0.650/ 29.6%	0.520/ 29.4%	0.346/ 27.75%
FLAD	**0.952/ 100%**	**0.949/ 100%**	**0.950/** 99.9%	**0.947/** 99.7%	**0.952/** 99.7%	**0.951/ 100%**

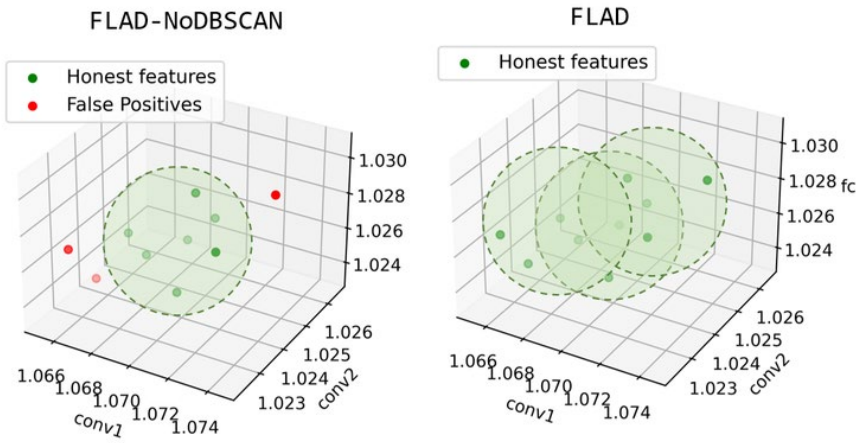


Figure 4. Comparison of feature vector partitioning by FLAD-NoDBSCAN and FLAD

The results in Table 2 show that FLAD performs better than the three variants. Because the server data set is small, and the server model has fewer training rounds per round than the client's local model. Therefore, FLAD-withServer directly weights and aggregates server model

parameters and honest client parameters together to obtain a global model with reduced accuracy. In FLAD-NoDBSCAN, due to the non-IID problem of client data, there are small differences in feature extraction vector groups between honest parameters, which may be ignored by direct use of range partitioning, as shown in Figure 4. Therefore, the malicious detection effect of FLAD is better than FLAD-NoDBSCAN. In FLAD-NoFCNN, because local model parameters belong to high-dimensional vectors, the DBSCAN clustering algorithm is directly used to divide high-dimensional local parameters without using feature extraction model for dimensionality reduction. Honesty and malice will bring huge errors.

To sum up, FLAD outperforms the other three ablation variants and that both DBSCAN and neural network feature extraction models play an integral role in the performance improvement of FLAD.