Exploiting Layerwise Feature Representation Similarity For Backdoor Defence in Federated Learning

Kane Walter[1](✉), Surya Nepal[1,2](✉), and Salil Kanhere[1](✉)

[1] University of New South Wales, Kensington, NSW 2033, Australia
{kane.walter,salil.kanhere}@unsw.edu.au
[2] CSIRO Data61, Marsfield, NSW 2122, Australia
surya.nepal@data61.csiro.au

Abstract. Federated learning is an emerging paradigm for distributed machine learning that enables clients to collaboratively train models while maintaining data privacy. However, this approach introduces vulnerabilities, notably the risk of backdoor attacks where compromised models may perform normally on clean data but maliciously on poisoned inputs. A range of defences has been proposed in the literature based on robust aggregation, differential privacy or certified robustness and clustering/trust score-based approaches. In this work, we introduce FedAvgCKA, a novel defence mechanism that leverages the learned representations of neural networks to distinguish between benign and malicious submissions from clients. We demonstrate the effectiveness of FedAvgCKA across various federated learning scenarios and datasets, showcasing its ability to maintain high main task accuracy and significantly reduce backdoor attack success rates even in non-iid settings.

Keywords: federated learning · backdoor attack · centered kernel alignment

1 Introduction

Federated Learning (FL) is a distributed machine learning technique that allows clients to train a neural network collaboratively [24]. In FL, clients train models on local data (stored on client devices) and then share the model weights with a central server. The central server generates a central model by aggregating the model weights received from the clients. An iterative process of client training and aggregation occurs until training converges, and then the central model is deployed for its designated task. The main advantage of FL is data privacy, as the clients share the trained models rather than raw training data. FL also reduces the computational costs associated with neural network training as clients train in parallel [24]. FL is an emerging technology but has found applications in several key areas, credit risk management [1], cardiac event prediction [6], mobile keyboard prediction [9,17] and healthcare informatics [35].

J. Garcia-Alfaro et al. (Eds.): ESORICS 2024, LNCS 14985, pp. 354–374, 2024.
https://doi.org/10.1007/978-3-031-70903-6_18

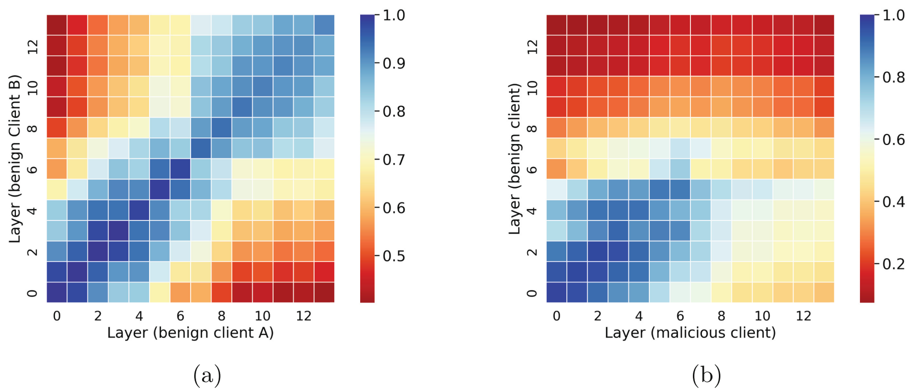

Fig. 1. Visualization of layerwise CKA similarity scores between models from two benign clients (left panel) and between benign and malicious clients (right panel) in a federated learning system. Benign clients exhibit high similarity in early layers and relatively high similarity in later layers, converging towards a unified feature space. For malicious clients executing a backdoor attack, early layers show similarity to benign clients, but deeper layers diverge. FedAvgCKA detects and excludes malicious clients based on this dissimilarity.

Despite its advantages, the privacy-preserving nature of FL prevents inspection of training data and model weights stored on client devices. This presents an attack surface that can be exploited to achieve poisoning attacks [16]. In general, poisoning attacks involve manipulating training data and/or model weights to influence model convergence toward an attacker-specified objective. Untargeted poisoning attacks aim to degrade the quality of the central model [12]. This type of attack is easily detected due to a sharp decline in model quality metrics (e.g., test accuracy) when the attack begins. Targeted poisoning attacks induce the central model to operate incorrectly in specific situations, which makes them harder to detect [16]. In this paper, we focus on FL backdoor attacks [3,31,34], which are a form of targeted poisoning attack and define a model that has been successfully attacked as a backdoored model. Generally, a backdoored central model will operate normally on clean test examples but predict attacker-chosen targets when given attacker-chosen test examples (i.e., poisoned examples). Since the server can't detect the poisoning process on the client side and given that backdoored models function correctly on clean inputs, they are a particularly devastating attack that deserves attention from the research community.

There are two main strategies used to defend against backdoor attacks in FL. The first strategy relies on detecting and discarding/down-weighting questionable model submissions from clients. Some examples include robust aggregation [5,25,36], anomaly detection using cosine distance [13] or clustering [26,29] and using trust metrics [8]. The drawback with these approaches is the need for strong assumptions about the attack methodology, data distribution among clients and distribution of model weight vectors. In real-world settings, these assumptions are unlikely to be met. The second strategy aims to limit the effect of malicious model updates using Differential Privacy (DP) [3,30] or certified robustness [33]. This strategy is effective against a broader set of threats because no assumptions about the adversary strategy are required. Unfortunately, this approach tends to reduce the quality of the central model by a significant amount as a trade-off for good backdoor robustness.

In this work, we develop a new backdoor defence based on the learned representations of a neural network. Our defence analyzes the behaviour of submitted client models rather than the raw values found in the client-submitted weight vectors. Specifically, at the beginning of training, the model trainer collects a small, clean test dataset for the learning task. During each round of training, this dataset is used as input for each client-submitted model. The internal network representation (i.e. activation values at an internal layer) for each input sample is obtained. The similarity of the stored representations is then determined using centered kernel alignment (CKA) [18]. Figure 1 provides a qualitative example of what our defence aims to detect using CKA. The clusters in the representation of the benign client are well-defined and separable. In contrast, the representation obtained from the malicious model exhibits tight packing and overlap. Such differences are irreconcilable through simple transformations and hints that internal representations learned in malicious models differ significantly from those in benign models. We call our defence FedAvgCKA for the remainder of this work and show that detecting malicious models based on their learned representation is an effective way to mitigate backdoor attacks in FL.

Goal and Contributions. Our contributions can be summarized as follows:

1. We propose FedAvgCKA, a novel backdoor defence method in federated learning that utilizes the learned representation of client models, marking a departure from traditional methods dependent on analysing client-submitted weight vectors.
2. FedAvgCKA is evaluated against current state-of-the-art backdoor attacks, demonstrating its capability to safeguard federated learning systems across various attack scenarios.
3. Our extensive testing validates the robustness of FedAvgCKA in diverse settings, including different attacker counts, data heterogeneity levels, and poisoning ratios, showcasing its adaptability to real-world federated learning challenges.

2 Background

In this section, we provide the necessary background for this work introducing basics of FL, backdoor attacks in FL and the theoretical background for centered kernel alignment (CKA).

Federated Learning. Federated learning is a distributed machine learning approach where a set of K clients collaboratively train a model that minimizes any finite-sum objective of the form:

$$\min_{w\in\mathbb{R}^d} f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w) = \sum_{k=1}^{K} \frac{|D_k|}{|D|} F_k(w) \tag{1}$$

where n is the number of training examples, $f_i(w) = l(x_i, y_i; w)$ is the training loss for training example (x_i, y_i), $|D|$ is the number of training examples in the system, $|D_k|$ is the number of local training examples held on client k and $F_k(w)$ is the local objective for client k.

The federated averaging algorithm [24] is commonly used to solve Eq. 1. Training is split into T communication rounds, and in each round t, the central server selects m clients to receive a copy of the current central model G^t. Each client produces a *local model* w_k^{t+1} by optimizing G^t on their local dataset d_k and returns w_k^{t+1} (or a *model update*, $w_k^{t+1} - G^t$) to the central server. An updated central model G^{t+1} is produced by aggregating the received client models. Eqs. 2 and 3 define the client and server training update rules under federated averaging, respectively:

$$w_k^{t+1} \leftarrow G^t - \eta g_k \tag{2}$$

$$G^{t+1} \leftarrow \sum_{k=1}^{m} \frac{|D_k|}{|D|} w_k^{t+1} \tag{3}$$

where η is the client learning rate and g_k is the local training gradient for client k.

Backdoor Attack. Backdoor attacks aim to produce a central model that predicts an attacker-specified label for any input that contains an attacker-specified trigger. In this work we deploy three well-known backdoor attacks from the literature and describe them below.

Model Poisoning. Bagdasaryan et al. [3] introduced the model poisoning attack which involves direct manipulation of model weights. Under model poisoning attacks, poisoned models (i.e., local models trained on poisoned data) are scaled up before being sent to the central server. The relatively larger norm of these models allows them to dominate the aggregation process. With as little as one compromised client in one round of training, the central model can be replaced with a poisoned model update [3].

Edge Case Backdoor. The edge case backdoor attack provides a higher level of stealth compared with model replacement due to the poisoned samples being drawn from the tail of the input distribution (i.e., rare, or underrepresented samples) [31]. Using digit recognition as an example, a poisoned sample could be a digit written in a different script to the client training distribution. Samples in ARDIS dataset [21] (digits written by priests in the 19th century) are within the overall distribution of handwritten digits but are in the tail of that distribution considering this is not the way digits are written in general. Most defences have difficulty mitigating this attack [31] because there is no backdoor trigger to defend against.

Distributed Backdoor Attack. The Distributed Backdoor Attack (DBA) more effectively exploits the structure of the FL architecture compared to other FL backdoor attacks [34]. This is achieved by distributing the trigger across several clients. As no single attack sample contains the complete trigger, it becomes impossible for the server to detect poisoned training samples, even when the trigger's shape is known. Like the edge case backdoor, DBA employs projected gradient descent. This technique, combined with the distributed trigger, significantly enhances the stealth of the attack, resulting in a more potent assault than the original model replacement attack.

2.1 Centered Kernel Alignment

CKA Overview. CKA was introduced to assess the similarity between internal representations, specifically activation matrices, of neural networks [18]. Let $X \in \mathbb{R}^{n \times p_1}$ and $Y \in \mathbb{R}^{n \times p_2}$ represent the activation matrices of $p \in p_1, p_2$ neurons for n examples, then CKA is the similarity measure $s(X, Y) \in [0, 1]$. The resulting value ranges from 0 (indicating completely dissimilar representations) to 1 (signifying identical representations). Thus, a higher CKA score denotes similar representations between X and Y, while a lower score points to their divergence.

CKA Definition. CKA collects the activation matrices of some internal network layer when n inputs are given to the network and then measures similarity. Concretely, a representation (i.e. X or Y defined above) comprises n activation matrices. The key insight provided by CKA is that one can measure the similarity between each pair of examples in the representations separately and then compare the similarity structures rather than comparing multivariate features of an example in the two representations [18]. CKA is based on the Hilbert-Schmidt Independence Criterion (HSIC), a method used to determine whether two kernels are statistically independent [15]. For $K_{i,j} = k(x_i, x_j)$ and $L_{i,j} = l(y_i, y_j)$ where k, l are kernels and for $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ the centering matrix, HSIC can be written as $HSIC(K, L) = \frac{1}{(n-1)^2} tr(KHLH)$ where $tr(X)$ is the trace of matrix X. CKA can then be computed as:

$$CKA(K, L) = \frac{HSIC(K, L)}{\sqrt{HSIC(K, K)HSIC(L, L)}} \tag{4}$$

CKA in Adversarial FL Settings. The utility of CKA as a diagnostic tool in FL is captured in the heat map visualizations seen in Fig. 1, which contrast the representation similarities of benign and malicious clients within a federated network.

3 FedAvgCKA Design

Motivation. This work focuses on detecting backdoored models in FL by analyzing the learned internal representation of client submissions. Unlike traditional methods that inspect [5] or manipulate [26,33] raw weight values, our approach scrutinizes the functional characteristics of models. Our approach focuses on the functional characteristics of models, which are more indicative of anomalous behavior than raw weights that can be subtly manipulated (e.g., through weight scaling [3,31,34]) without being detected. CKA's ability to capture linear and non-linear relationships in high-dimensional spaces makes it the perfect tool for segregating models with divergent internal representations.

Algorithm 1. Compute centered kernel alignment (CKA) score from activation matrices (X and Y) where I_n is the identity matrix, $\mathbf{11}_n$ is an $n \times n$ dimension matrix of ones, $tr()$ is the trace of the given matrix, and HSIC is the Hilbert-Schmidt Independence Criterion [15].

```
1:  Input: Activation matrices $X, Y$ from clients $i, j$
2:  Output: CKA similarity score
3:  procedure CKA($X, Y$)
4:      $n \leftarrow$ rows in $X$                                ▷ X, Y have same dimensions
5:      $H \leftarrow I_n - \frac{1}{n}\mathbf{11}^\top$                 ▷ Centering matrix
6:      $X \leftarrow XH$                                         ▷ Center X
7:      $Y \leftarrow YH$                                         ▷ Center Y
8:      $HSIC_{XY} \leftarrow \mathrm{tr}(XY) \cdot (n-1)^{-2}$
9:      $HSIC_{XX} \leftarrow \mathrm{tr}(XX) \cdot (n-1)^{-2}$
10:     $HSIC_{YY} \leftarrow \mathrm{tr}(YY) \cdot (n-1)^{-2}$
11:     return $\frac{HSIC_{XY}}{\sqrt{HSIC_{XX} \cdot HSIC_{YY}}}$
12: end procedure
```

3.1 Design Challenges

Activation Collection. In this work, we consider CKA [18] within the context of FL, so a decision on where in the system (i.e., on the server or client) the collection of activation matrices should occur needs to be made.

Client-Side Collection of Activation Matrices. Opting for client-side activation collection and $HSIC_{XX}$ calculation distributes computational workload but poses risks. Malicious clients might send manipulated activation matrices, along with poisoned model updates, compromising the system. Heterogeneous data distribution in FL may favor malicious submissions over benign ones, particularly in highly non-iid settings. Moreover, the availability of activation matrices and model weights raises data privacy concerns, potentially enabling reverse-engineering of local training data, leading to enhanced inference attacks in untrusted server scenarios.

Server-Side Collection of Activation Matrices. Deploying the entire FedAvgCKA process on the server has pros and cons. A significant advantage is heightened security: full control over the root dataset (see Sect. 4) reduces the risk of receiving manipulated activation matrices from clients. Using a consistent dataset to generate activation matrices enables accurate classification of models as benign or malicious, addressing concerns about internal representation discrepancies due to local data composition. However, this assumes access to a clean root dataset, a strong assumption in FL settings. Additionally, there's increased computational load on the server for obtaining pairwise similarity scores for all client submissions, particularly notable with more complex kernels like RBF.

For the remainder of this work, we will concentrate on the server-side collection of activations for FedAvgCKA. This decision is based on the need to maintain greater control over the dataset used to generate activation matrices and the desire for maximal robustness/privacy in the FL process despite the associated increased computational demands and the presumption of having a root dataset on the server.

Type of Kernel. In applying CKA to FL, an important consideration is the choice of kernel function. The kernel function determines the nature of the feature space in which the similarities between models are compared. Linear and RBF (Radial Basis Function) are the two commonly used kernels.

RBF Kernel. The RBF kernel, $k(x_i, x_j) = exp(-||x_i - x_j||_2^2/(2\sigma^2))$, is non-linear and can capture a broad range of data distributions. It maps the inputs

Table 1. Comparison of main task accuracy between standard federated learning (FL) and federated learning under FedAvgCKA (CKA) across different Dirichlet parameter settings. No malicious clients were included in the client pool for these experiments.

	MNIST		CIFAR10		FASHION	
Dirichlet (α)	*FL*	*CKA*	*FL*	*CKA*	*FL*	*CKA*
0.2	98.8	98.2	84.1	78.9	97.1	95.7
0.4	98.9	98.5	87.9	82.4	97.6	96.9
0.6	98.9	98.6	88.7	84.8	98.0	96.7
0.8	98.9	98.6	89.7	85.9	98.1	97.4
1.0	98.9	98.7	89.8	86.5	98.1	97.5

into higher-dimensional spaces, enabling the detection of complex and subtle patterns in the data (i.e. internal representations for FedAvgCKA). This can be particularly beneficial in FL settings where client data is heterogeneous, leading to highly divergent internal representations among client models. However, the RBF kernel comes with increased computational demands and complexity in tuning the bandwidth parameter σ, which controls the emphasis on the similarity of small distances in feature space over large ones.

Linear Kernel. The linear kernel, denoted as a simple dot product in the input space, stands out for its simplicity and computational efficiency. It is particularly effective where relationships in the data are predominantly linear. In the context of CKA, the linear kernel offers straightforward interpretability and is computationally less demanding than more complex kernels. This makes it a practical choice for large-scale FL applications, where computational efficiency is crucial. All things being equal, the more expressive RBF kernel would appear superior to the linear kernel due to the complexity of neural network feature space. However, Kornblith et al. [18] showed that both kernels yield similar results in most circumstances, suggesting that the linear kernel is a reliable option for a wide range of deep learning scenarios.

Based on these considerations, we have chosen to use the linear kernel for the remainder of this paper. Its advantageous balance of efficiency, simplicity, and robustness makes it a fitting choice for our scenario. Despite the RBF kernel's

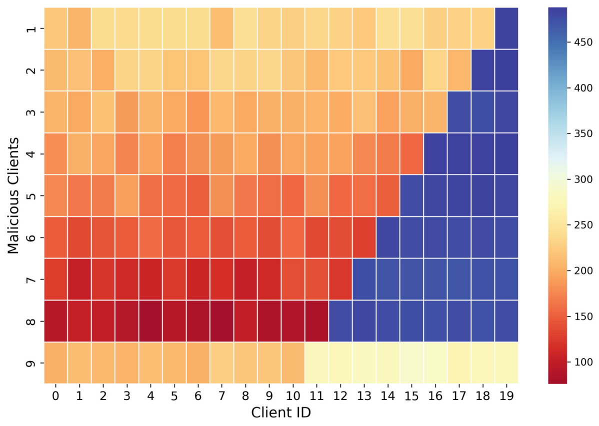

Fig. 2. Detection frequency of malicious clients across nine federated learning experiments each running over 500 training rounds. These experiments included 100 clients in total with 20 being selected for training in each round. The y-axis denotes how many malicious clients where included in each training round of a given experimental run, while the x-axis represents client IDs, with IDs 12–19 corresponding to the malicious clients in a given round. Deeper blue in the figure represents more frequent detection (Color figure online).

ability to manage complex, non-linear relationships, its computational complexity and sensitivity to the parameter σ render it less suitable for our purposes.

3.2 Implementation

A textual overview of FedAvgCKA is provided in this section that should be read in conjunction with Algorithm 1 which presents the core of the defence and detailed Algorithm 2 in Appendix A which shows where Algorithm 1 fits into the FL workflow. FedAvgCKA includes a pre-aggregation step that uses the CKA similarity score to determine which client submissions are malicious. The server obtains pairwise CKA similarity scores for m client submissions in round $t \leq T$ and discards the submissions least similar to all other submissions.

The steps outlined below overview Algorithm 1 and Algorithm 2. References to line numbers refer to lines of Algorithm 2. We now outline the steps of FedAvgCKA:

1. **Initialization step** (line 1): The process begins with the initialization of the global model G_0, setting the stage for the federated learning rounds.
2. **Federated Training Rounds** (lines 3 - 8): For each federated training round $t \leq T$, standard FL client local training [24] occurs in parallel:
   - A fraction C of clients K is randomly chosen for training and included in set S_t. Each client $k \in S_t$ trains locally on G_{t-1}, generating client submission w_t^k.
   - Client submission w_t^k sent back to the server.
3. **Activation Matrix Generation** (line 9): Activation matrices a_t^k are generated by the server by inputting the root dataset R into the received client submission and capturing activations at a user-specified layer. If model updates (i.e. $w_t^k - G_{t-1}$) are returned rather than full model weights, the server adds w_t^k back to G_{t-1} to reconstruct the client-side model before inputting R.
4. **CKA-Based Anomaly Detection** (lines 13 - 23): On the server, the CKA procedure (Algorithm 1) is used to compute pairwise similarity scores between the activation matrices from different clients. This is achieved by centering the activation matrices, calculating the HSIC values and returning the CKA similarity score (normalized HSIC). Client submissions with the lowest average similarity scores are presumed malicious and removed from W.
5. **Model Aggregation** (line 24): The new global model G_t is generated using the standard FedAvg aggregation method [24] applied to the remaining client submissions in W.

4 Experimental Setup

We implemented and trained all neural network models using PyTorch (v2.0) [27]. Open-source code obtained from GitHub was used to implement DBA, model poisoning and edge case backdoor attacks. Malicious clients were included in each training round using the fixed-frequency method [30]. Baselines including Multi-Krum, FoolsGold, FLAME, and FLTrust were selected to represent an array of different defence styles and their popularity in the federated backdoor literature.

Table 2. Results showing backdoor success (BA) and main task accuracy (MA) for standard federated averaging (FedAvg) and FedAvgCKA. The MNIST, CIFAR10, and Fashion MNIST datasets were subjected to DBA [34], Edge case [31], and model poisoning [3] attacks, respectively.

		FedAvg		FedAvgCKA	
Attack	**Dataset**	*BA*	*MA*	*BA*	*MA*
DBA	MNIST	100.0	98.7	1.3	98.4
Edge case	CIFAR10	100.0	88.0	6.2	84.4
Model poisoning	FASHION	100.0	97.3	2.4	96.3

Federated Learning System. We simulated a federated system with 100 heterogeneous client devices, following previous work in the FL backdoor attack literature [8,26]. Each training round involved a random 20% subset of clients, with malicious client percentages varying from 0 to 50% across experiments (i.e., at worst, 50% of clients selected for training in round t were forced to be malicious). Model updates were aggregated using federated averaging over 500 rounds to maximize attacker chance of inserting backdoor. Clients selected for training in round t conducted one local epoch with a batch size of 64. Client-side learning rate was set at 0.1, while the server learning rate remained fixed at 1.0 throughout. Experiments were conducted with varying batch sizes and Dirichlet hyperparameters to simulate different federated environments.

A root dataset R is required on the server for FedAvgCKA to operate (see Sect. 3.2 and Algorithm 2 in Appendix A). Also, like defences from the literature that are based on similarity measures/outlier exclusion [5,26], FedAvgCKA is expected to be robust up to 50% malicious clients in each round. Therefore, FedAvgCKA excludes 50% of client submissions in each training round to account for the worst-case situation.

Datasets. We used three benchmark datasets, MNIST [22], CIFAR10 [19], and Fashion MNIST [32]. MNIST contains 70,000 28×28 grayscale digit images, facilitating fundamental learning and model integrity evaluation. CIFAR10, comprising 60,000 32×32 color images across ten classes, offers challenges due to image variability. Fashion MNIST, with 70,000 grayscale images of fashion items, replaces handwritten digits with clothing, testing model discrimination abilities. These datasets were chosen for their popularity in the FL backdoor attack literature, varying degrees of complexity, and relevance to practical applications. The diversity of the datasets allows for a comprehensive assessment of the vulnerability of FL to backdoor attacks with FedAvgCKA deployed as a defence.

Evaluation Metrics. We evaluate the effectiveness of FedAvgCKA using a set of standard quality metrics. We consider Backdoor Accuracy (BA), which measures the accuracy of the backdoor task. BA represents the number of times

Table 3. Comparison of attack success rate (BA) and main task accuracy (MA) across several defence mechanisms: Multi-Krum [5], FoolsGold [13], FLTrust [8], FLAME [26], and FedAvgCKA.

	CIFAR10		MNIST		FASHION	
Defences	*BA*	*MA*	*BA*	*MA*	*BA*	*MA*
FedAvg (No attack)	0.0	88.1	0.0	98.7	0.0	98.0
FedAvg (No defence)	100.0	87.9	100.0	98.1	100.0	97.6
Multi-Krum	95.1	83.8	98.9	95.5	98.1	94.2
FoolsGold	90.0	82.8	98.0	96.4	97.8	96.4
FLTrust	11.9	84.2	3.4	98.5	2.8	97.5
FLAME	10.1	78.4	2.2	96.9	2.7	94.4
FedAvgCKA	10.4	84.9	0.8	98.7	1.5	97.8

the attacker-specified target was predicted when the model was given a poisoned test dataset. The adversary aims to maximize BA, while an effective defence should minimize it.

We assess main task accuracy (MA), which measures the accuracy of the main learning task. MA represents the number of times the ground truth label was predicted when the model was given a clean test dataset. Both the adversary and the defender seek to maximize MA but for different reasons. The adversary requires high MA for stealth purposes, whereas benign clients require it for quality. A robust defence preferable has no impact on MA.

5 Experimental Results

Benign Convergence. Like other backdoor defences that focus on outlier exclusion [5,13], FedAvgCKA faces the potential risk of excluding benign models. This is a significant issue in FL because a key principle of training is including the information contained in heterogeneous data shards. Therefore, assessing the convergence behavior of benign FL systems under FedAvgCKA is crucial. Table 1 compares main task accuracy between standard FL and FedAvgCKA in scenarios with no malicious clients and varying data heterogeneity.

In our experiments, both FL and FedAvgCKA demonstrated broadly similar final model quality across varying levels of α, although FedAvgCKA exhibited a consistent slight decrease in accuracy compared to standard FL. This decrease was more noticeable at lower α values, especially for CIFAR10. However, as data distribution became more homogeneous, this disparity reduced, aligning with the expected behavior of anomaly-detection defences in non-iid settings.

These findings suggest that while FedAvgCKA might occasionally exclude benign model updates during training, its impact on the overall system performance is not substantial and FedAvgCKA ultimately attains performance comparable to standard FL by the end of training.

Table 4. Performance of FedAvgCKA against backdoor attacks (BA) across varying percentages of attackers within the system alongside the main task accuracy (MA).

	MNIST		CIFAR10		FASHION	
Attackers (%)	*BA*	*MA*	*BA*	*MA*	*BA*	*MA*
0	0.0	98.6	0.0	85.4	0.0	97.3
10	0.9	98.7	0.3	85.4	1.6	97.4
20	0.8	98.7	0.2	85.6	1.8	97.6
30	0.8	98.7	10.4	84.9	1.5	97.8
40	18.7	98.7	86.9	84.0	69.0	97.8
50	89.7	98.7	91.8	84.2	90.0	97.0

Defence Effectiveness. Table 2 demonstrates the effectiveness of FedAvgCKA across various datasets while maintaining high main task accuracy. As expected, for all baseline datasets FedAvg was not robust obtaining 100% attack success. On the other hand, FedAvgCKA markedly lowered the attack success to 1.3% for MNIST, 6.2% for CIFAR10, and 2.4% for Fashion MNIST, with only slight reductions in main task performance. These results support the hypothesis that malicious clients are learning significantly different internal representations to benign clients and that FedAvgCKA can exploit this to exclude them.

Defence Specificity. We conducted nine experiments to evaluate how precise FedAvgCKA malicious client detection is and report the results in Fig. 2. In each experiment, twenty clients were selected for training in each round (out of a pool of 100 clients). Each client was assigned a round ID (IDs 12–19 were reserved for malicious clients), and FedAvgCKA classified each as benign or malicious with a count being kept across 500 rounds of training. Taking a concrete example from Fig. 2, we can see that when 8 malicious clients were included in each training round, they could clearly be delineated from the 12 benign clients as the cells corresponding to ID numbers 12–19 are deep blue and the ID number 0-11 being dark red.

Comparison to Baseline. Table 3 presents a comparative analysis of FedAvgCKA against several baselines defences from the literature under DBA [34]. Unsurprisingly, without any defence, the system is entirely vulnerable to attacks, with a backdoor attack success of 100% across all datasets. Multi-Krum and FoolsGold struggled to mitigate attacks effectively due to the highly heterogeneous data split ($\alpha = 0.4$) and relatively low number of attackers in this experiment (20% client pool was compromised for this experiment), respectively. FLAME's norm clipping/noise based approach successfully mitigated the attack but at the cost of main task performance, indicating a compromise between security and accuracy in line with previous findings on DP-based defences [3]. FLTrust was found to be a robust defence, maintaining low BA while preserving

MA, likely due to its trust-based strategy that leverages a representative dataset for accurate model update evaluation. Similarly, FedAvgCKA also displayed good defensive capability, keeping BA rates considerably low with minimal impact on MA across all datasets.

Number of Malicious Clients. Table 4 shows how FedAvgCKA performs under varying levels of system compromise. For MNIST, main task accuracy consistently stays above 98.6%, but backdoor attack success rate escalates rapidly from 0.8% to 89.7% when the proportion of attackers rises from 30% to 50%. Fashion MNIST follows a similar pattern, maintaining over 97% of main task accuracy, yet suffering from increasing backdoor attack success rates with more attackers. CIFAR10 maintained main task accuracy around 85% but showed increased susceptibility to backdoor attacks as the proportion of malicious clients rises. These results suggest that while FedAvgCKA effectively preserves main task ask accuracy, its defensive efficacy decreases as more attackers are included in the system. As the proportion of attackers in the system grows toward 50%, it becomes much harder for outlier detection based defences such as FedAvgCKA to detect malicious submissions. Therefore, malicious updates are inadvertently included in the federated aggregation process and attack success rate increases.

Table 5. Performance of FedAvgCKA at mitigating backdoor attacks (BA) across varying degrees of data heterogeneity, specified by the Dirichlet distribution parameter (α).

	MNIST		CIFAR10		FASHION	
Dirichlet (α)	*BA*	*MA*	*BA*	*MA*	*BA*	*MA*
0.2	32.1	98.4	44.8	76.3	26.7	95.0
0.4	1.3	98.5	8.2	82.0	2.3	96.4
0.6	4.9	98.6	4.9	84.0	1.4	96.8
0.8	1.1	98.7	3.1	84.7	1.6	97.0
1.0	1.0	98.7	4.8	85.8	1.5	97.3

Data Distribution. We evaluated the performance of FedAvgCKA under varying degrees of data heterogeneity, as characterized by the Dirichlet distribution parameter (where lower α is more heterogeneous) and report the results in Table 5. For all three datasets, FedAvgCKA was robust down to approximately $\alpha = [0.3, 0.4]$, particularly MNIST and Fashion-MNIST. This means that FedAvgCKA is robust in all but the most severe environments of data heterogeneity. However, these results highlight a fundamental challenge for FedAvgCKA and similar outlier detection defences in FL. In environments with diverse data sets, differentiation between normal and anomalous client submissions becomes exceedingly difficult [5,13].

Table 6. Backdoor attack success rate (BA) and main task accuracy (MA) for FedAvgCKA under varying poisoning ratio. We found that FedAvgCKA is most effective at moderate to high poisoning ratios, as malicious models learn representations sufficiently different from benign models.

	MNIST		CIFAR10		FASHION	
Ratio (%)	*BA*	*MA*	*BA*	*MA*	*BA*	*MA*
12.5	35.3	98.7	44.7	85.9	20.1	97.2
25.0	1.0	98.7	42.9	86.9	1.7	97.5
50.0	1.0	98.7	36.2	86.8	2.1	97.6
75.0	0.8	98.7	24.0	87.4	1.8	97.6

Poisoning Ratio. We evaluated how FedAvgCKA performs under different levels of poisoned training samples on malicious clients (poisoning ratio) and report the results in Table 6. The results revealed lower poisoning ratios were more effective at compromising the system than higher ratios. For MNIST, a backdoor attack success rate of 35.3% was observed at a 12.5% poisoning ratio, likely due to the poisoned models' representations closely mimicking those of benign models. However, as the ratio increased, backdoor attack success rates declined sharply. This trend was also evident in the Fashion MNIST dataset. The CIFAR10 results showed a gradual decrease in backdoor attack success as the poisoning ratio increased. Edge-case backdoors use poison samples from outside the client data distribution which produces gradients that target weights in the model that are not altered by in-distribution data. This increases backdoor persistence as the poisoned weights are less likely to be corrected by training on in-distribution data in future rounds. Even though FedAvgCKA can detect such attackers, once the backdoor is inserted, it is harder to remove by simply excluding attackers from future training rounds.

Size of Root Dataset. FedAvgCKA relies on a root dataset stored on the server to generate representations of models submitted by clients. This section focuses on assessing the impact of varying the size of this root dataset on the effectiveness of FedAvgCKA. Figure 3 depicts the fluctuation in backdoor attack success as the root dataset size ranges from a small scale (4 samples) to a larger one (256 samples). Our observations indicate that even a relatively small root dataset, comprising as few as 8 samples, can provide adequate model representation, as evidenced by the low attack success rates observed with datasets ranging from 8 to 256 samples. However, when the dataset falls below 8 samples, there is insufficient dissimilarity between the model representations, hindering FedAvgCKA's ability to identify and exclude malicious submissions effectively.

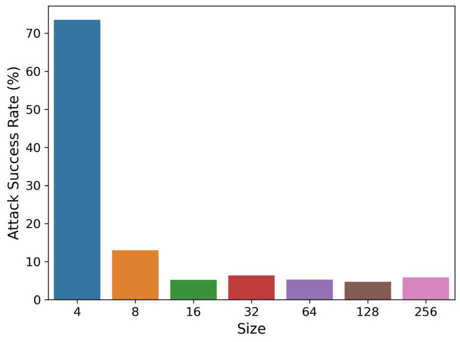

Fig. 3. Variation of backdoor attack success rate with the size of the root dataset used in FedAvgCKA, ranging from 4 to 256 samples. This figure shows that the root dataset can be as small as 16 samples for good separability between representations learned by benign and malicious models.

Table 7. Comparison of the computational cost of various backdoor defences and FedAvgCKA. Parameters d, n, and k refer to the number of weights in the model, the number of clients selected for training in each round and the size of the root dataset, respectively.

Method	Computation at server	Memory cost to client
FedAvg [24]	$\mathcal{O}(d)$	$\mathcal{O}(d)$
FedAvgCKA	$\mathcal{O}(n^2k^2d + k^3)$	$\mathcal{O}(d)$
Multi-Krum [5]	$\mathcal{O}(n^2d)$	$\mathcal{O}(d)$
FLAME [26]	$\mathcal{O}(n^2d)$	$\mathcal{O}(d)$
FLTrust [8]	$\mathcal{O}(knd)$	$\mathcal{O}(d)$

Training Costs. Table 7 outlines the computational demands of various FL methods, with FedAvgCKA standing out as the most resource intensive. This is primarily due to the matrix multiplications required for calculating the CKA similarity metric (Algorithm 1), which are performed pairwise across all client submissions in each training round (Algorithm 2). While FedAvgCKA enhances robustness against backdoor attacks, its higher computational demands may limit its suitability for resource-constrained central servers. However, these requirements can be mitigated by using a smaller root dataset (e.g., as few as 16 samples were effective in our experiments) and optimizing matrix operations through hardware enhancements.

Limitations of FedAvgCKA. Despite the promising results demonstrated in this study, several limitations of our approach must be acknowledged. Firstly, our method requires a clean root dataset, which may not always be available in real-world scenarios. However, our experiments indicate that the root dataset can be relatively small. As shown in Fig. 3, a dataset with as few as 16 samples can still provide adequate model representation, thereby alleviating the stringent requirement for a large clean root dataset. Secondly, the computational cost associated with our approach is considerable. The resource-intensive nature of the computations can render our method impractical for deployment on edge devices and/or systems that select many clients for training in each round. Lastly, while our approach offers improvements over previous proposals, these gains are relatively modest. The incremental nature of the performance enhancement may not justify the additional complexity and resource expenditure in some applications. Addressing these limitations will be crucial in advancing the applicability and effectiveness of our method in diverse and real-world settings and will be investigated in future iterations of the proposed defence.

6 Related Work

In this section, we survey backdoor attack techniques and proposed mitigation strategies.

Backdoor Attacks. Backdoor attacks modify models to perform well on their main task and an additional malicious task. Gu et al. [16] pioneered deep neural network backdoors by imprinting visual trigger patterns on mislabeled training examples. Chen et al. [10] proposed invisible trigger patterns blended into images. Physical backdoors use objects in images as triggers [10,16], posing persistent concerns in outsourced model training through transfer learning [20,31].

In FL, semantic backdoors occur through model poisoning [3,4], scaling malicious updates to replace the central model. Xie et al. [34] demonstrated distributed trigger-based attacks in FL. Wang et al. [31] introduced edge case backdoors, forcing models to misclassify seemingly easy test examples from extreme data distribution tails.

Robust Aggregation Defences. Robust aggregation methods like Krum [5] address security concerns in federated learning (FL) by selecting client updates with minimal distances to others, assuming the fraction of malicious clients is below a threshold. Algorithms like Bulyan [25] and coordinate-wise methods (median and trimmed-mean [36]) offer similar guarantees but depend on the server seeing all client submissions and assume iid data. In non-iid environments, these algorithms struggle to distinguish malicious updates, leading to the exclusion of benign contributions.

Data Pre-processing Defences. Input preprocessing defenses aim to neutralize visual triggers in images. Liu et al. used an autoencoder to reconstruct images, making triggers ineffective [23]. Februus improved this by using computer vision to locate and in-paint likely trigger regions [11]. Gao et al. superimposed patterns on inputs and filtered those with low entropy in predicted classes, indicating a potential trigger [14].

Fine Tuning Defence. To mitigate backdoors, modifying the model to suppress trigger activity is effective. Liu et al. [23] re-trained models with benign examples in outsourcing contexts. Qiao et al. [28] adapted by re-training models based on trigger distributions rather than specific patterns. Liu et al. [23] also integrated neuron pruning to remove dormant neurons on clean inputs alongside re-training.

Anomaly Detection Defence. FoolsGold [13] identifies colluding attackers using cosine similarity but sacrifices many valid updates and central model quality. FLAME integrates HDBSCAN [7] for client clustering and applies Differential Privacy methods [2,3,30]. DeepSight [29] also uses HDBSCAN and enhances it with a voting-based model filtering method combining model similarity and a meta classifier for backdoor defense.

Trust metrics. FLTrust [8] scales client updates by trust score, effective for detecting backdoor attacks but violating FL's assumption of server visibility into client distributions. Baffle [2] delegates backdoor detection to clients, utilizing local data for trigger detection, yet requiring the full trigger in client data and assuming iid data are its main shortcomings.

7 Conclusion

In this paper, we introduced FedAvgCKA, a novel defence mechanism designed to enhance the robustness of federated learning systems against backdoor attacks. Unlike existing methods, FedAvgCKA exploits the layerwise feature representation dissimilarity between benign and malicious clients to detect and mitigate backdoor attacks. Our comprehensive evaluations across multiple datasets reveal that FedAvgCKA, even when faced with sophisticated adversarial strategies, effectively maintains the integrity of the global model.

Acknowledgment. This work was supported by the Cyber Security Research Centre Limited, whose activities are partially funded by the Australian Government's Cooperative Research Centres Programme.

A Appendix A: FedAvgCKA Algorithm

Algorithm 2. FedAvgCKA Algorithm where T is the number of federated training rounds, C is the fraction of clients chosen for training in each round, K is the set of system clients, *choose()* selects a random subset of clients from the client pool, *clientUpdate()* is the standard FL local training procedure [24], *getActivations()* obtains activation matrix using root dataset (R), w_t^k and a_t^k are client submission and activation matrix respectively for client k in round t, *addItem()* adds items to lists, *average()* maintains the average CKA score for client $i \in S_t$, *sort*() sorts CKA scores in ascending order and *aggregate*() is the standard FedAvg aggregation [24]. On line 22 we discard 50% of the round t clients.

```
1: Initialize global model $G_0$
2: procedure FedAvgCKA($G_0$)
3:     for each round $t = 1, 2, \ldots, T$ do
4:         $m \leftarrow \max(C \cdot |K|, 1)$
5:         $S_t \leftarrow$ choose(K, m)                      ▷ $S_t \subseteq K$, $|S_t| = m$
6:         $W, A \leftarrow [], []$
7:         for each client $k \in S_t$ do
8:             $w_t^k \leftarrow$ clientUpdate($k, G_{t-1}$)          ▷ $w_t^k$ to server
9:             $a_t^k \leftarrow$ getActivations($w_t^k, R$)          ▷ On server
10:            addItem(W, $w_t^k$)
11:            addItem(A, $a_t^k$)
12:        end for
13:        scores = {}                                ▷ Dictionary/hash table
14:        for $X_i$ in A do
15:            for $Y_j$ in A do
16:                score $\leftarrow$ CKA($X, Y$)                ▷ Algorithm 1
17:                average(scores[i], score)
18:                average(scores[j], score)
19:            end for
20:        end for
21:        S = sort(scores)
22:        S = S[m/2:]                                ▷ Determine anomalous models
23:        $W = W \setminus S$                                ▷ Exclude anomalous models
24:        $G_t \leftarrow$ aggregate($W$)
25:    end for
26:    return $G_T$
27: end procedure
```

References

1. Utilization of fate in risk management of credit in small and micro enterprises. https://www.fedai.org/cases/utilization-of-fate-in-risk-management-of-credit-in-small-and-micro-enterprises/

2. Andreina, S., Marson, G.A., Möllering, H., Karame, G.: Baffle: backdoor detection via feedback-based federated learning. In: 41st IEEE International Conference on Distributed Computing Systems, ICDCS 2021, Washington DC, USA, July 7-10, 2021, pp. 852–863. IEEE (2021)
3. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. In: Chiappa, S., Calandra, R. (eds.) The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]. Proceedings of Machine Learning Research, vol. 108, pp. 2938–2948. PMLR (2020)
4. Bhagoji, A.N., Chakraborty, S., Mittal, P., Calo, S.B.: Analyzing federated learning through an adversarial lens. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 634–643. PMLR (2019)
5. Blanchard, P., Mhamdi, E.M.E., Guerraoui, R., Stainer, J.: Machine learning with adversaries: byzantine tolerant gradient descent. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA, pp. 119–129 (2017)
6. Brisimi, T.S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I.C., Shi, W.: Federated learning of predictive models from federated electronic health records. Int. J. Med. Inform. **112**, 59–67 (2018)
7. Campello, R.J.G.B., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD 2013. LNCS (LNAI), vol. 7819, pp. 160–172. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37456-2_14
8. Cao, X., Fang, M., Liu, J., 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. The Internet Society (2021)
9. Chen, M., Mathews, R., Ouyang, T., Beaufays, F.: Federated learning of out-of-vocabulary words. CoRR **abs/1903.10635** (2019). http://arxiv.org/abs/1903.10635
10. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. CoRR **abs/1712.05526** (2017). http://arxiv.org/abs/1712.05526
11. Doan, B.G., Abbasnejad, E., Ranasinghe, D.C.: Februus: input Purification Defense Against Trojan Attacks on Deep Neural Network Systems, pp. 897–912. ACM, New York, NY, USA (2020)
12. Fang, M., Cao, X., Jia, J., Gong, N.Z.: Local model poisoning attacks to byzantine-robust federated learning. In: Capkun, S., Roesner, F. (eds.) 29th USENIX Security Symposium, USENIX Security 2020, August 12–14, 2020, pp. 1605–1622. USENIX Association (2020)
13. Fung, C., Yoon, C.J.M., Beschastnikh, I.: The limitations of federated learning in sybil settings. In: Egele, M., Bilge, L. (eds.) 23rd International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2020, San Sebastian, Spain, October 14–15, 2020, pp. 301–316. USENIX Association (2020)
14. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S.: STRIP: a defence against trojan attacks on deep neural networks. In: Balenson, D. (ed.) Proceedings

of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019, pp. 113–125. ACM (2019)

15. Gretton, A., Fukumizu, K., Teo, C.H., Song, L., Schölkopf, B., Smola, A.J.: A kernel statistical test of independence. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S.T. (eds.) Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3–6, 2007, pp. 585–592. Curran Associates, Inc. (2007)
16. Gu, T., Liu, K., Dolan-Gavitt, B., Garg, S.: BadNets: evaluating backdooring attacks on deep neural networks. IEEE Access **7**, 47230–47244 (2019)
17. Hard, A., et al.: Federated learning for mobile keyboard prediction (2018). https://arxiv.org/abs/1811.03604
18. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.E.: Similarity of neural network representations revisited. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 3519–3529. PMLR (2019)
19. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009). https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
20. Kurita, K., Michel, P., Neubig, G.: Weight poisoning attacks on pretrained models. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 2793–2806. Association for Computational Linguistics (2020)
21. Kusetogullari, H., Yavariabdi, A., Cheddad, A., Grahn, H., Hall, J.: ARDIS: a Swedish historical handwritten digit dataset. Neural Comput. Appl. **32**(21), 16505–16518 (2019). https://doi.org/10.1007/s00521-019-04163-3
22. LeCun, Y., Cortes, C., Burges, C.: MNIST handwritten digit database. ATT Labs **2** (2010). http://yann.lecun.com/exdb/mnist
23. Liu, Y., et al.: Trojaning attack on neural networks. In: 25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18–21, 2018. The Internet Society (2018)
24. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Singh, A., Zhu, J. (eds.) Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282. PMLR, Fort Lauderdale, FL, USA (2017)
25. Mhamdi, E.M.E., Guerraoui, R., Rouault, S.: The hidden vulnerability of distributed learning in byzantium. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 3518–3527. PMLR (2018)
26. Nguyen, T.D., et al.: FLAME: taming backdoors in federated learning. In: Butler, K.R.B., Thomas, K. (eds.) 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022, pp. 1415–1432. USENIX Association (2022)
27. Paszke, A., Gross, S., et al.: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035. Curran Associates, Inc. (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

28. Qiao, X., Yang, Y., Li, H.: Defending neural backdoors via generative distribution modeling. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 14004–14013 (2019)
29. Rieger, P., Nguyen, T.D., Miettinen, M., Sadeghi, A.: Deepsight: mitigating backdoor attacks in federated learning through deep model inspection. In: 29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24–28, 2022 (2022)
30. Sun, Z., Kairouz, P., Suresh, A.T., McMahan, H.B.: Can you really backdoor federated learning? CoRR **abs/1911.07963** (2019). http://arxiv.org/abs/1911.07963
31. Wang, H., et al.: Attack of the tails: yes, you really can backdoor federated learning. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual (2020)
32. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. CoRR **abs/1708.07747** (2017). http://arxiv.org/abs/1708.07747
33. Xie, C., Chen, M., Chen, P., Li, B.: CRFL: certifiably robust federated learning against backdoor attacks. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event. Proceedings of Machine Learning Research, vol. 139, pp. 11372–11382. PMLR (2021)
34. Xie, C., Huang, K., Chen, P., Li, B.: DBA: distributed backdoor attacks against federated learning. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020 (2020)
35. Xu, J., Glicksberg, B.S., Su, C., Walker, P.B., Bian, J., Wang, F.: Federated learning for healthcare informatics. J. Heal. Inform. Res. **5**(1), 1–19 (2021)
36. Yin, D., Chen, Y., Ramchandran, K., Bartlett, P.L.: Byzantine-robust distributed learning: towards optimal statistical rates. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 5636–5645. PMLR (2018). http://proceedings.mlr.press/v80/yin18a.html