Universal Detection of Backdoor Attacks via Density-Based Clustering and Centroids Analysis

Wei Guo, Benedetta Tondi, *Member, IEEE*, and Mauro Barni, *Fellow, IEEE*

***Abstract*— We propose a Universal Defence against backdoor attacks based on Clustering and Centroids Analysis (CCA-UD). The goal of the defence is to reveal whether a Deep Neural Network model is subject to a backdoor attack by inspecting the training dataset. CCA-UD first clusters the samples of the training set by means of density-based clustering. Then, it applies a novel strategy to detect the presence of poisoned clusters. The proposed strategy is based on a general misclassification behaviour observed when the features of a representative example of the analysed cluster are added to benign samples. The capability of inducing a misclassification error is a general characteristic of poisoned samples, hence the proposed defence is attack-agnostic. This marks a significant difference with respect to existing defences, that, either can defend against only some types of backdoor attacks, or are effective only when some conditions on the poisoning ratio or the kind of triggering signal used by the attacker are satisfied. Experiments carried out on several classification tasks and network architectures, considering different types of backdoor attacks (with either clean or corrupted labels), and triggering signals, including both global and local triggering signals, as well as sample-specific and source-specific triggers, reveal that the proposed method is very effective to defend against backdoor attacks in all the cases, always outperforming the state of the art techniques.**

***Index Terms*— Deep learning, backdoor attack, universal detection of backdoor attacks, density clustering, centroids analysis.**

I. Introduction

DEEP Neural Networks (DNNs) are widely utilised in many areas such as image classification, natural language processing, and pattern recognition, due to their outstanding performance over a wide range of domains. However, DNNs are vulnerable to both attacks carried out at test time, like the creation of adversarial examples [1] and training time [2]. These vulnerabilities limit the application of DNNs in security-sensitive scenarios, like autonomous vehicle, medical diagnosis, anomaly detection, video-surveillance and many others. One of the most serious threats comes from backdoor attacks [3], [4], [5], [6], according to which a portion of the training dataset is poisoned to induce the model to learn a malevolent behaviour. At test time, the *backdoored* model works as expected on normal data, however, the hidden backdoor and the malevolent behaviour are activated when the network is fed with an input containing a so-called triggering signal, only known to the attacker.

Backdoor attacks can be categorised into two classes: *corrupted-label* and *clean-label* attacks [7]. In the first case, the attacker can modify the labels of the poisoned samples, while in the latter case, the attacker does not have this capability. Hence, in a clean-label backdoor attack, the poisoned samples are correctly labelled, i.e., the content of a poisoned sample is consistent with its label. For this reason, clean-label attacks [8], [9] are more stealthy and harder to be-detected than corrupted-label attacks.

Many methods have been proposed to defend against backdoor attacks. Following the taxonomy introduced in [7], the defences can be categorised into three different classes based on the knowledge available to the defender and the level at which they operate: *sample-level*, *model-level*, and *training-dataset-level* defences. Sample-level defences are applied after the model has been deployed in an operative environment, and rely on the inspection of the input sample to reveal the possible presence of a triggering signal. With model-level defences, instead, the network is inspected before its deployment to detect the possible presence of a backdoor. Finally, defences working at the training-dataset-level assume that the defender can access and inspect the dataset used to train the network to look for suspicious (poisoned) samples. The CCA-UD defence introduced in this paper belongs to this last category.

A. Related Works

Most of the defence methods working at the training-dataset level rely on clustering and on the analysis of the feature representations or activation patterns. One of the earliest and most popular approach is the Activation Clustering (AC) method [10]. By focusing on corrupted-label attacks, the AC method analyses the feature representation of the samples of each class of the training dataset, and clusters them, in a reduced dimensionality space, via the K-means ($K = 2$) algorithm [11]. Under the hypothesis that a benign class tends to form a homogeneous cluster in the feature space, and by noticing that when K-means is forced to identify two clusters in the presence of only one homogeneous cluster, it tends to split it into two equally-sized clusters, the data samples of a

Manuscript received 19 June 2023; revised 5 October 2023; accepted 29 October 2023. Date of publication 1 November 2023; date of current version 23 November 2023. This work was supported in part by the Italian Ministry of University and Research through the PREMIER Project and in part by the China Scholarship Council (CSC) under Grant 201908130181. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Valeria Loscri. *(Corresponding author: Wei Guo.)*

The authors are with the Department of Information Engineering and Mathematics, University of Siena, 53100 Siena, Italy (e-mail: wei.guo.cn@outlook.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIFS.2023.3329426, provided by the authors.

Digital Object Identifier 10.1109/TIFS.2023.3329426

class are judged to be poisoned on the basis of the relative size of the two clusters identified by K-means. If the size of the two clusters is similar, the class is considered to be benign, otherwise, the class is judged to be poisoned. Finally, AC labels the samples of the smallest cluster as poisoned samples. The method works under the assumption that the fraction of poisoned samples (hereafter referred to as the poisoning ratio) in a poisoned class is lower than the number of benign samples. On the other hand, given that K-means does not work well in the presence of clusters with very unbalanced sizes [12], AC does not perform well when the poisoning ratio is very small, as it often happens in practice with corrupted-label attacks, thus limiting the applicability of AC.

Xiang et al. [13] presented the Cluster Impurity (CI) method, which works under the assumption that the triggering signal used by the attacker can be removed by an average filter. Specifically, given the training samples of one class, CI analyses their feature representation and groups the samples into K clusters by exploiting the Gaussian Mixture Model (GMM) algorithm [14]. The number of clusters K is determined by the Bayesian Information Criterion (BIC) [15]. To determine whether one cluster includes poisoned samples or not, CI average filters all the cluster samples and observes if the classification of these samples change. Thanks to this intuition, CI can still work when the number of poisoned samples in the poisoned class is larger than the number of benign samples. For the same reason, CI only works against corrupted-label attacks, given that in a clean-label setting the prediction made by the network on the filtered samples would not change. Moreover, the applicability of CI is limited to specific kinds of triggering signals, that is, triggers with high-frequency components, that can be removed via a low pass filter (like an average filter).

In 2021, Tang et al. [16] proposed a so-called Statistical Contamination Analyser (SCAn), that relies on the Expectation Maximisation (EM) algorithm to decompose the representation of an image/object into an identity and a variation part. They argue that while the representations of samples from different (benign) classes have different identities, they share the same intra-variation distribution. Based on this intuition, SCAn estimates the intra-variation from a benign dataset, and uses it to define a statistical hypothesis test to judge whether a given class is contaminated (H1) or not (H0). If H1 occurs, SCAn splits the representations into two groups via Linear Discriminant Analysis (LDA). A limitation of SCAn is that it fails to defend against sample-specific attacks, as shown in [17], likely due to a different intra-variability for the poisoned class when sample-specific triggers are considered.

To overcome the limitation of [16] and [17] proposed a system, named Beatrix, to detect poisoned samples via anomaly detection. Similarly to [16] and [17] exploits the availability of a small amount of benign data. In particular, it computes the Gram matrix to derive class statistics from the benign samples. The deviation of the feature points of the input sample from the Gramian feature representation for the class is measured to detect the anomalies induced by the poisoned samples. However, since the entries of Gram matrix can be viewed as the inner product of two vectors, it suffers from the curse of dimensionality, and so when the feature dimensionality is large the performance of Beatrix drops.[1]

While there are other defences working at the training-dataset level, most of them assume that the defender has some additional, often unrealistic, knowledge about the backdoor attack. For instance, the method introduced in [18], and its strengthened version described in [19], exploit singular value decomposition (SVD) [20] to reveal the anomalous samples contained in the training dataset, assuming that an upper-bound of the fraction of poisoned samples is known. Shan et al. [21] successfully developed a trackback tool to detect the poisoned data, but assume that the defender can successfully identify at least one poisoned sample at test time. Moreover, some defences only target a specific kind of backdoor attack. For instance, [22] aims at defending against clean-label backdoor attacks by exploiting feature collision. Recently, [23] and [24] proposed two methodologies to train a backdoor-free model from a poisoned dataset, exploiting randomised smoothing [25] and adversarial training [26]. The same goal is accomplished in [27] where a training methodology is proposed to learn deconfounded representations for reliable classification, by relying on the minimization of the mutual information between the to-be-trained model and backdoored model (obtained by training for some epochs on the poisoned dataset). Finally, [28] utilises self-attention to purify the backdoored model, exploiting the relationship in the structural information between shallow and deep layers characterising benign models.

B. Contribution

In view of the limitations, in terms of general applicability of the defences proposed so far, we introduce a universal training-dataset-level defence, named CCA-UD, which can reveal the presence of poisoned data in the training dataset regardless of the approach used to embed the backdoor (corrupted- or clean-label), the size and the shape of the triggering signal, the use of fixed, sample- or class-specific triggers, and the percentage of poisoned samples. To obtain such a noticeable result, we observe that clustering alone is not sufficient to tell apart poisoned and benign samples. In fact, due to intra-class variability, also benign classes may (and, in fact, are) split in various clusters. In this case, however, the clusters are all benign, i.e., containing benign samples, and hence have to be detected as such. As a matter of fact, most previous defences based on clustering recognise this fact and analyse the clusters in some way to distinguish benign and poisoned samples, however, they do so by targeting a specific backdoor attack, or a specific class of attacks, thus failing to achieve a universal defence capable of detecting the presence of poisoned samples for all the vast variety of attacks proposed so far. On the contrary, CCA-UD relies on the general observation that samples belonging to a poisoned cluster have some common features that, when added to benign samples, cause a misclassification error. With these ideas in mind, the novel contribution of CCA-UD can be summarised

[1] Our experiments reveal that Beatrix fails when the dimensionality of the feature representation is large, i.e., >9000).

as follows: i) adoption of a clustering algorithm, namely Density-based Spatial Clustering of Application with Noise (DBSCAN) [29], capable of isolating poisoned samples from benign ones; and ii) introduction of a new active strategy to check if the residual features of the various clusters induce a general misclassification behaviour when they are added to the features of benign samples.

CCA-UD is applied immediately after the model has been trained and aims at detecting if the training data contains poisoned samples causing the generation of a backdoor into the trained model. Similarly to SCAn [16] and Beatrix [17], it assumes that the defender has access to a small set of benign samples for each class in the input domain of the model, while AC and CI do not require this knowledge).

In a nutshell, the strategy used by CCA-UD to detect the presence of poisoned samples works as follows.

For every class in the training set, we apply clustering in the latent feature spaces, splitting each class into multiple clusters. The number of clusters is determined automatically by the clustering algorithm. If clustering works as expected, benign and poisoned samples are grouped into different clusters. To decide whether a cluster is poisoned or not, we first recover an average representation of the cluster by computing the cluster's *centroid*. For a poisoned cluster, the centroid will likely contain the representation of the triggering signal in the feature space. Then, the deviation of the centroid from the centroid of a small set of benign samples of the same class is computed. The deviation vector computed in this way is finally added to the feature representations of the benign samples of the other classes. If such an addition causes a misclassification of (a large portion of) the benign samples, the corresponding cluster is judged to be poisoned.

We have tested the validity and universality of CCA-UD, by evaluating its performance against many different backdoor attacks carried out against DNN-based classifiers, considering different classification tasks, namely, MNIST, traffic sign, fashion clothes, CIFAR10 and YouTubeFace, two poisoning strategies, i.e., corrupted- and clean-label poisoning, and several triggering signals, namely two global patterns - a ramp and a sinusoidal signal - a square local pattern, and also source-specific and sample-specific triggers. Our experiments show that CCA-UD provides an effective defence against backdoor attacks in all scenarios, outperforming the state-of-the-art methods.

The rest of the paper is organised as follows: in Section II, we provide the basic notation used in the paper and the background. In Section III, we introduce our defence threat model and present the CCA-UD defence. Section IV describes the experimental methodology we followed to evaluate the performance of the proposed defence, and Section V shows the corresponding results of the experiments. Finally, we conclude our paper in Section VI.

II. Notation and Background

In a backdoor attack, the attacker, say Eve, aims at embedding a backdoor into a model by poisoning some samples of the training set. Specifically, we assume that the task addressed by the model targeted by the attack is a classification task. Let t denote the target class of the attack. Eve corrupts part of the training set, in such a way that, at test time, the backdoored model works normally on benign data, but misclassifies the input sample to the target class t, if the triggering signal υ is present within it.

Let us denote the clean training dataset by $D_{tr} = \bigcup_i D_{tr,i}$, where $D_{tr,i}$ is the set of samples belonging to class i, $i = 1, \ldots, l$, and l denotes the number of classes. Then, each class set is defined as $D_{tr,i} = \{(x_j, i), j = 1, \ldots, |D_{tr,i}|\}$, where the pair (x_j, i) indicates the j-th sample of class i and its label. Similarly, we use the notation D_{ts} and $D_{ts,i}$ for the test dataset. Eve corrupts D_{tr} by merging it with a poisoned set $D^p = \{(\tilde{x}_j, t), j = 1, \ldots, |D^p|\}$, where $\tilde{x}_j$ denotes the j-th poisoned sample, containing the trigger υ, labeled as belonging to class t. The poisoned dataset is indicated as $D^{\alpha}_{tr} = D_{tr} \cup D^p$ (with α defined later). Then, for the class targeted by the attack we have $D^{\alpha}_{tr,t} = D_{tr,t} \cup D^p$, while for the other classes, we have $D^{\alpha}_{tr,i} = D_{tr,i}$ $(i \neq t)$. The fraction $\alpha = |D^p|/|D_{tr,t} \cup D^p|$ indicates the poisoning ratio used by the attacker to corrupt the training set.

As we said, D^p can be generated by either corrupting the labels of the poisoned samples or not. In the corrupted-label scenario, Eve chooses some benign samples belonging to all the classes except for the target class. Then, she poisons each sample-label pair with a poisoning function $\mathcal{P}$, obtaining the poisoned samples $(\tilde{x}_j, \tilde{y}_j = t) = \mathcal{P}(x_j, y_j \neq t)$. The symbol $\tilde{x}_j$ is the poisoned sample including the triggering signal υ. In the clean-label case, Eve cannot corrupt the labels, so she chooses some benign samples belonging to the target class and generates the poisoned samples as $(\tilde{x}_j, \tilde{y}_j = t) = \mathcal{P}(x_j, y_j = t)$. In contrast with the corrupted-label case, now $\mathcal{P}()$ embeds υ into x_j to generate $\tilde{x}_j$, but keeps the label intact.

Arguably, defending against corrupted-label attacks is easier, since mislabeled samples can be more easily identified upon inspection of the training dataset, observing the inconsistency between the content of the samples and their labels. In contrast, clean-label attacks are more stealthy and more difficult to detect. However, clean-label attacks are more difficult to implement since they require that a larger portion of the dataset is corrupted [5], [30].

We denote the DNN model trained on D^{α}_{tr} by F^{α}. Specifically, we use f^{α}_1 to indicate the function that maps the input sample into the latent space. In this paper, we assume that f^{α}_1 includes a final ReLu layer [31], so that its output is a non-negative vector. Hence, $f^{\alpha}_1(x)$ is the feature representation of x, and f^{α}_2 is used to denote the classification function. Formally, $F^{\alpha}(x) = f^{\alpha}_2(f^{\alpha}_1(x))$. Finally, the dimension of the feature representation is denoted by d.

Table I summarises the main notation used in the paper.

A. Density-based Clustering

In this paragraph, we describe the Density-based Spatial Clustering of Application with Noise (DBSCAN) [29] algorithm used by CCA-UD. DBSCAN splits a set of points into K clusters and possibly few outliers, where K is automatically determined by counting the areas with high sample density. Specifically, given a point 'A' of the set, DBSCAN

TABLE I

LIST OF SYMBOLS

D_{tr}, $D_{tr,i}$	Training dataset, i-th class subset of training dataset
D_{ts}, $D_{ts,i}$	Test dataset, i-th class subset
D_{val}, D^i_{val}	Validation dataset, i-th class subset
D^p	Set of poisoned samples
$\mathcal{P}()$, t, α	Poisoning function, target class, class poisoning ratio
D^α_{tr}, $D^\alpha_{tr,i}$	Poisoned training dataset, its i-th class subset
$F^\alpha(f^\alpha_1, f^\alpha_2)$	Model trained on D^α_{tr} (feature mapping, classification)
$f^\alpha_1(x)$	Representation of x (d-dim) extracted from F^α
C^k_i	k-th cluster of points in D^i_{val}
MR^k_i	Misclassification ratio in favor of class i, for cluster k
P_i (B_i)	Set of samples detected as poisoned (benign) for class i
GP_i, GB_i	Ground-truth poisoned and benign samples for class i
PC	Poisoned class
BC_P	Benign class of poisoned training dataset
BC_B	Benign class of benign training dataset

counts the number of neighbours (including 'A' itself) within a distance ϵ from 'A'. If the number of neighbours is larger than or equal to a threshold $minPts$, 'A' is defined to be a *core* point and all points in its ϵ-neighbourhood are said to be *directly reachable* from 'A'. If a point, say 'B', of the reachable set is again a core point, all the points in its ϵ-neighbours are also *reachable* from 'A'. Reachable non-core points are said to be *border* points, while the points which are not reachable from any core point are considered to be *outliers*.

To define a cluster, DBSCAN also introduces the notion of density-connectedness. We say that two points 'A' and 'B' are density-connected if there is a point 'C', from which both 'A' and 'B' are reachable (hence 'C' must be a core point). A clusters is defined as a group of points satisfying the following two properties: i) the points within a cluster are mutually density-connected; ii) any point directly-reachable from some point of the cluster, it is part of the cluster. The intuition behind DBSCAN is to define the clusters as dense regions separated by border points. The number of dense regions found in the set automatically determines the number of clusters K. More information about the exact way the clusters are found and the (in-)dependence of DBSCAN on the initial point 'A' used to start the definition of core and reachable points, are given in the original paper [29].

The performance of DBSCAN is determined by the choice of the parameters involved in its definition, that is $minPts$ and ϵ, whose setting depends on the problem at hand. The setting of these parameters in CCA-UD is discussed in Section IV-C, while their impact of the performance on CCA-UD is evaluated in the ablation study reported in the supplementary material.

We choose to adopt a density-based clustering method as the backbone of CCA-UD, since density-based clustering is known to work well also in the presence of clusters with unbalanced size [32], and because it provides an automatic way to determine the number of clusters.[2]

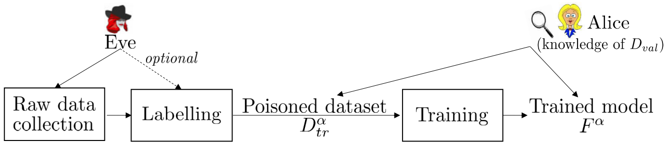


Fig. 1. Threat model.

III. THE PROPOSED UNIVERSAL DEFENCE

A. Defence Threat Model

The threat model considered in this work is illustrated in Fig. 1. The attacker Eve interferes with the data collection process, by poisoning a fraction α of the training dataset, possibly modifying the labels of the poisoned samples. Alice, plays the role of the trainer, defining the model architecture, learning algorithm, and hyperparameters, and training the model. Alice also plays the role of the defender: she inspects the training dataset and the deployed model to detect the possible presence of poisoned samples in the training set. The exact goal, knowledge and capabilities of the defender are detailed in the following.

Defender's goal: Alice aims at revealing the presence of poisoned samples in the training dataset D^α_{tr}, if any, and identify them.[3] Upon detection of the poisoned samples, Alice may remove them from the training set and use the clean dataset to train a sanitised model.

Formally, the core of the CCA-UD defence consists of a detector, call it $det()$, defined as follows. For every subset $D^\alpha_{tr,i}$ of the training dataset D^α_{tr}, $det(D^\alpha_{tr,i}) = (P_i, B_i)$, where P_i and B_i are the sets with the samples in class i judged by $det()$ to be, respectively, poisoned and benign. Extending the above functionality to all the classes in the input domain of the classifier, we have $det(D^\alpha_{tr}) = \{(P_i, B_i), i = 1, \ldots, l\}$. Clearly, for a non-poisoned dataset, ideally, we should have $P_i = \emptyset \quad \forall i$, since no sample is poisoned in any class.

Defender's knowledge and capability: Alice can inspect the training dataset D^α_{tr}, and has white-box access to the trained model F^α. Moreover, Alice has a small benign validation dataset D_{val}, with a small number of non-poisoned samples of every class. This is a requirement common to other methods like SCAn, Beatrix (while AC and CI do not require it).

B. The Proposed CCA-UD Defence

CCA-UD consists of two main blocks: *feature clustering* and *Poisoned Cluster Detection (PCD)*, shown in Fig. 2 and detailed in Sections III-B.1 and III-B.2.[4]

Feature clustering relies on the DBSCAN algorithm [29]. DBSCAN splits a set of points into K clusters and possibly few outliers, where K is automatically determined by counting the areas with high sample density. We refer to [29] for more information on the DBSCAN method. The performance of

[2] DBSCAN is one of most popular density-based clustering algorithms, other choices, like OPTICS [33] and HDBSCAN [34]) would work as well.

[3] For sake of simplicity, we use the notation D^α_{tr} for the training set under inspection, even if, prior to inspection, we do not know if the set is poisoned or not. For benign dataset we simply have $\alpha = 0$.

[4] The code implementing CCA-UD is available at the address https://github.com/guowei-cn/CCA_UD-universal-training-level-defence.git

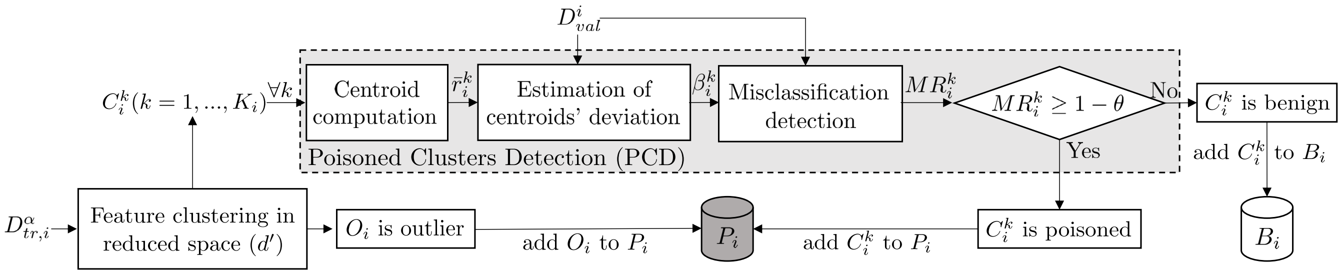


Fig. 2. Workflow of the CCA-UD defence.

DBSCAN is affected by the choice of the parameters involved in its definition, that is $minPts$ (the threshold on the number of neighbours used to define core points, which determines the dense regions constituting the clusters) and ϵ (distance defining the neighbourhood), whose setting depends on the problem at hand. The influence of such parameters on CCA-UD and the way we set them are described in Section IV-C. We decided to use density-based clustering since it works well also in the presence of clusters with unbalanced size [32], and because it provides an automatic way to determine the number of clusters.[5]

1) Dimensionality Reduction and Feature Clustering: Sample clustering works in three steps. In the first step, for every class i, we compute the feature representations of all the samples in $D^\alpha_{tr,i}$, namely $\{f^\alpha_1(x_j), x_j \in D^\alpha_{tr,i}\}$. Vector $f^\alpha_1(x_j)$ is a d-dim vector. Secondly, we reduce the dimension of the feature space from d to d' via Uniform Manifold Approximation and Projection (UMAP) [35]. Finally, we apply DBSCAN to split $D^\alpha_{tr,i}$ into multiple clusters $C^k_i(k = 1, \dots, K_i)$. In addition to clusters, DBSCAN (may) also return a number of outliers. The set with the outlier samples, referred to as O_i, is directly added to P_i. The outlier ratio for the class i is denoted by $\zeta_i = \frac{|O_i|}{|D^\alpha_{tr,i}|}$. With the hyperparameters (d', $minPts$ and ϵ) we have chosen, ζ_i is usually very small (see Table S1 reported in the supplementary material).

Notably, in [36], the separability of poisoned and benign features (after PCA reduction) in different layers is investigated, to understand if some layers are more effective than others for the discrimination, and a method is proposed to find the layer where poisoned samples and benign samples are most distinguishable. According to our experiments, however, when UMAP [35] algorithm is used for dimensionality reduction, poisoned samples can be distinguished well from benign samples in all layers, and CCA-UD achieves similar performance regardless of the layer where the analysis is carried out. A visualisation of feature separability after PCA and UMAP is shown in the supplementary material (Section II).

We observe that the dimensionality reduction is exploited only to run DBSCAN, while in the other steps the full feature space is considered. Reducing the dimensionality of the features before applying DBSCAN permits to reduce the time complexity of the algorithm and at the same time avoids curse of dimensionality problems [37], occurring when clustering is applied in high-dimensional spaces (see the experiments reported in the supplementary material - Section IV - supporting this claim).

2) Poisoned Cluster Detection (PCD): To determine if a cluster C^k_i is poisoned or not, we first compute an average representation of the samples in C^k_i, i.e., the cluster's centroid. Then, we check whether the centroid contains a feature component that causes a misclassification in favour of class i when added to the features of benign samples of the other classes. More specifically, we first calculate the centroid of C^k_i as $\bar{r}^k_i = E[f^\alpha_1(x_j)|x_j \in C^k_i]$, where $E[\cdot]$ denotes component-wise sample averaging. Vector $\bar{r}^k_i$ is a d-dimensional vector.[6] Then, we compute the deviation of $\bar{r}^k_i$ from the centroid of class i computed on a set of benign samples:

$$\beta^k_i = \bar{r}^k_i - E[f^\alpha_1(x_j)|x_j \in D^i_{val}], \tag{1}$$

where D^i_{val} is the i-th class of the benign set D_{val}.

Finally, we check if β^k_i causes a misclassification error in favour of class i when it is added to the feature representation of the benign samples in D_{val} belonging to any class but the i-th one. The corresponding misclassification ratio is computed as follows:

$$MR^k_i = \frac{\sum_{x_j \in D_{val}/D^i_{val}} \mathbb{1}\{f^\alpha_2\left(\delta(f^\alpha_1(x_j) + \beta^k_i)\right) \equiv i\}}{|D_{val}/D^i_{val}|}, \tag{2}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function (outputting 1 when the condition is satisfied and zero otherwise), D_{val}/D^i_{val} represents the validation dataset excluding the samples from class i, and δ is a ReLu operator that ensures that $f^\alpha_1(x_j) + \beta^k_i$ is a correct vector in the latent space (see Section II).

For a given threshold θ, if $MR^k_i \geq 1-\theta$,[7] the corresponding C^k_i is judged to be poisoned and its elements are added to P_i. Otherwise, the cluster is considered benign and its elements are added to B_i. Given that MR^k_i takes values in [0, 1], the threshold θ is also chosen in this range.

3) Expected Behaviour of CCA-UD for Clean- and Corrupted-Label Attacks: An intuition of the idea behind CCA-UD and the reason why the detection of poisoned samples works for both corrupted- and clean-label attacks is given in the following. Let us focus first on the clean-label

[5] DBSCAN is one of the most popular density-based clustering algorithms, other choices, like OPTICS [33] and HDBSCAN [34]) would work as well.

[6] We remind that, although clustering is applied in the reduced-dimension space, the analysis of the clusters is performed in the full features space.

[7] We defined the threshold as $1-\theta$ to ensure that TPR and FPR increase with the growth of θ as for AC and CI, so to ease the comparison between the various defences.

attack scenario. If cluster C_i^k is poisoned, the centroid $\bar{r}_i^k$ contains the features of the trigger in addition to the feature of class i. Then, arguably, the deviation of the centroid from the average representation of class i is a significant one. Ideally, subtracting to $\bar{r}_i^k$ the average feature representation of the i-th class, obtaining β_i^k, isolates the trigger features. The basic idea behind CCA-UD is that the trigger features in β_i^k will cause a misclassification in favour of class i when added to the features of benign samples of the other classes. On the contrary, if cluster C_i^k is benign, the centroid $\bar{r}_i^k$ approximates the average feature representation of the i-th class and then β_i^k has a very small magnitude. In this case, β_i^k accounts for normal intra-class fluctuation of the features and its addition to benign samples is not expected to induce a misclassification.

Similar arguments, with some noticeable differences, hold in the case of corrupted-label attacks. As before, for a benign cluster C_i^k, the centroid $\bar{r}_i^k$ approximates the average feature representation of the i-th class and then β_i^k corresponds to minor intra-class variations. In the case of a poisoned cluster C_i^k, the cluster now includes mislabeled samples of the other classes (different from i) containing the triggering signal. In this way, the cluster representative contains features of the original classes in addition to the features of the triggering signal. Note that even if the clustering algorithm splits the poisoned samples across more than one cluster, the deviation vector β_i^k of poisoned clusters will contain the features of the triggering signal (possibly in addition to features accounting for *difference* between the original class i and the target class t). Given that the network has been trained to *recognise* the triggering signal as a distinguishing feature of class t, once again, the addition of the deviation vector to benign samples is likely to cause a misclassification in favour of class t.

The situation is pictorially illustrated in Fig. 3 for a 3 dimension case, in the case of a clean-label attack (a similar picture can be drawn in the corrupted label case). Class '3' corresponds to the poisoned class. Due to the presence of the backdoor, the poisoned samples are characterised by a non-null feature component along the z direction. Due to the presence of such a component, the backdoored network classifies those samples in class '3'. On the contrary, benign samples lie in the x-y plane. When CCA-UD is applied to the samples labelled as Class '3', DBSCAN identifies two clusters, namely C_3^1 and C_3^2, where the former is a benign cluster and the latter is a poisoned cluster containing a non-null z-component. When PCD module is applied to C_3^1 (left part in the figure), the deviation from the set of benign samples of class i (β_3^1), has a small amplitude and lies in the x-y plane, hence when β_3^1 is added to the other clusters it does not cause a misclassification error. Instead, when the PCD module is applied to C_3^2 (right part in the figure), the deviation vector (β_3^2) contains a significant component in the z direction, causing a misclassification when added to the benign samples in D_{val}^1 and D_{val}^2.

It is worth stressing that CCA-UD indirectly exploits a general behaviour induced by backdoor attacks, that also works for sample-specific attacks when the samples are poisoned by applying a specific processing (e.g. a deformation/warping [38]). In this case, the backdoored network associates the traces of the specific processing applied to the input samples to one or more peculiar features (fingerprint), and uses such an evidence to misclassify these samples as belonging to the target class (it is possible that, in this case, the features of the poisoned samples are characterised by a larger variability with respect to the case in which the attack relies on a fixed triggering signal).

4) Discussion: We observe that the universality of CCA-UD essentially derives from the use of DBSCAN and from the generality of the proposed strategy for PCD. Firstly, in contrast to K-means, DBSCAN can handle unbalanced clusters, ensuring good performance also when the poisoning ratio α is small or when the number of poisoned samples is larger than the number of benign samples. Secondly, CCA-UD also works when the class samples have large intra-variability, since the ultimate decision on the presence of a cluster with poisoned samples is made is made by observing the *corrupting capabilities* of the cluster samples. In addition, in the presence of large intra-class variability, DBSCAN groups the data of a benign class into multiple clusters (a large K_i, $K_i > 2$, is estimated by DBSCAN), that are then detected as benign clusters. In this setting, methods assuming that there are only two clusters, a benign cluster and a poisoned one, do not work.

Finally, we observe that thanks to the fact that K_i is directly estimated by DBSCAN, in principle our method can also work in the presence of multiple triggering signals [39], [40]. In this case, the samples poisoned by different triggers would cluster in separate clusters, that would all be detected as poisoned by CCA-UD.

IV. Experimental Methodology

A. Evaluation Metrics

The performance of the backdoor attacks is evaluated by providing the accuracy of the backdoored model F^α on benign data and the success rate of the attack when the model is tested on poisoned data. The two metrics we use to do that are:

- Accuracy (ACC), measuring the probability of a correct classification of benign samples, and calculated as follows:

$$ACC = \sum_{i=1}^{l} \sum_{x_j \in D_{ts,i}} \mathbb{1}\{F^\alpha(x_j) \equiv i\}/|D_{ts}|; \qquad (3)$$

- Attack Success Rate (ASR), measuring the probability that the triggering signal υ activates the desired behaviour of the backdoored model F^α, computed as follows:

$$ASR = \frac{1}{|D_{ts}/D_{ts,t}|} \sum_{x_j \in D_{ts}/D_{ts,t}} \mathbb{1}\{F^\alpha(\mathcal{P}(x_j, \upsilon)) \equiv t\}, \qquad (4)$$

  where $D_{ts}/D_{ts,t}$ is the test dataset excluding the samples from class t.

In our experiments, a backdoor attack is considered successful when ASR is greater than 0.90 and the ACC of the poisoned model is similar to that of a model trained over benign samples (in our experiments such a difference is smaller than 0.01). To measure the performance of the defence algorithms, we

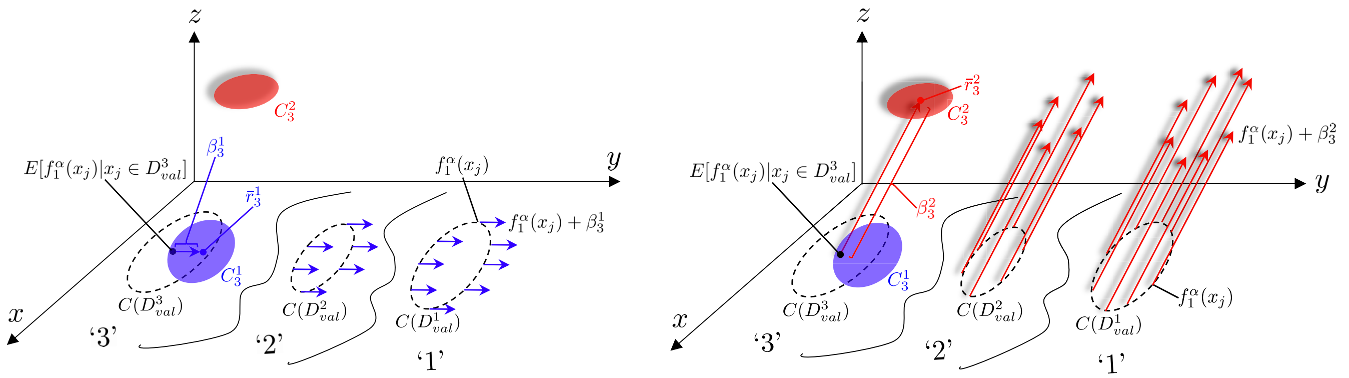


Fig. 3. Simplified illustration of PCD (clean-label case). For class '3' as the poisoned class, DBSCAN identifies two clusters: C_3^1 (benign) and C_3^2(poisoned). When PCD is applied to C_3^1 (left part), the deviation from the set of benign samples of class i ($C(D_{val}^3)$) has a small amplitude and lies on x-y plane, which cannot cause misclassification when added to the benign samples in D_{val}^1 and D_{val}^2. Instead, the deviation vector of C_3^2 (right part), containing a significant component in the z direction, can cause misclassification.

measure the True Positive Rate (TPR) and the False Positive Rate (FPR) of the defence. Actually, when i corresponds to a benign class, there are no poisoned samples in $D_{tr,i}^\alpha$ and only the FPR is computed. More formally, let GP_i (res. GB_i) define the set of ground-truth poisoned (res. benign) samples in $D_{tr,i}^\alpha$. We define the TPR and FPR on $D_{tr,i}^\alpha$ as follows:

$$TPR(D_{tr,i}^\alpha) = \frac{|P_i \cap GP_i|}{|GP_i|}, \; FPR(D_{tr,i}^\alpha) = 1 - \frac{|B_i \cap GB_i|}{|GB_i|}. \tag{5}$$

Given that benign classes may exist for both poisoned and benign datasets,[8] we need to distinguish between these two cases. Hence, we introduce the following definitions:

- Benign Class of Benign dataset (BC_B): a class of a clean dataset. In this case $\alpha = 0$ and $D_{tr,i}^\alpha$ includes only benign samples.
- Benign Class of Poisoned dataset (BC_P): a benign class of a poisoned dataset, that is, a class in a poisoned dataset different from the target class. Also in this case, $D_{tr,i}^\alpha$ includes only benign samples.

The difference between BC_B and BC_P is that in the former case F^α is a clean model, while in the latter it is backdoored. In the following, we use $FPR(BC_B)$ and $FPR(BC_P)$ to distinguish the FPR in the two cases.

Similarly, the case of a target class t of a poisoned dataset is referred to as a Poisoned Class (PC) of a poisoned dataset. In this case, $D_{tr,i=t}^\alpha$ includes both poisoned and benign samples, then we compute and report $TPR(PC)$ and $FPR(PC)$. TPR and FPR depend on the choice of the threshold θ. Every choice of threshold defines a different operating point of the detector. In order to get a global view of the performance of the tested systems, then, we also provide the AUC value, defined as the Area Under the Curve obtained by varying the value of the threshold and plotting TPR as a function of FPR.

According to the definitions in Eq. (5), the false positive and true positive rates are computed for each cluster. For the sake of simplicity, we will often report average values. For the case of benign clusters of a benign dataset, the average value, denoted by $\overline{FPR}(BC_B)$, is calculated by averaging over all the classes of the benign training dataset. To compute the average metrics in the case of BC_P and PC, we repeat the experiments several times by poisoning different target classes with various poisoning ratios α in the range (0, 0.55] for every target class, and by using the poisoned datasets to train the backdoored models.[9] Then, the average $\overline{FPR}(BC_P)$ is computed by averaging the performance achieved on non-target classes of all the poisoned training datasets. For the PC case, the average metrics $\overline{FPR}(PC)$, $\overline{TPR}(PC)$ and $\overline{AUC}$ are computed by averaging the values measured on the target classes of the poisoned training datasets. We also measured the average performance achieved for a fixed poisoned ratio α, by varying only the target class t. The notation $\overline{FPR}_\alpha(BC_P)$, $\overline{FPR}_\alpha(PC)$, $\overline{TPR}_\alpha(PC)$, $\overline{AUC}_\alpha$ is used in this case to highlight the dependence on α.

B. Network Tasks and Attacks

To validate the effectiveness of CCA-UD, we carried out extensive experiments considering the following classification tasks and attacks.

1) MNIST Classification: In this set of experiments, we trained a model to classify the digits in the MNIST dataset, which includes $n = 10$ digits (classes) with 6000 binary images per class. The size of the images is 28 × 28. The architecture used for the task is a 4-layer network [41]. The feature representation of dimensionality 128 is obtained from the input of the final fully-connected layer.

Regarding the attack setting, three different backdoor attacks have been considered, as detailed below. For each setting, the training dataset is poisoned by considering 16 poisoning ratios α chosen in (0, 0.55]. For each α, 10 different poisoned training datasets are generated by choosing different classes as the target class.

- Corrupted-label attack, with a 3 × 3 pixel trigger (3 × 3 *corrupted*): the backdoor is injected by adding a 3 × 3 pixel pattern to the corrupted samples, as shown

[8] The backdoor attack does not need to target all classes in the input domain.

[9] Only successful backdoor attacks are considered to measure the performance in the various cases.

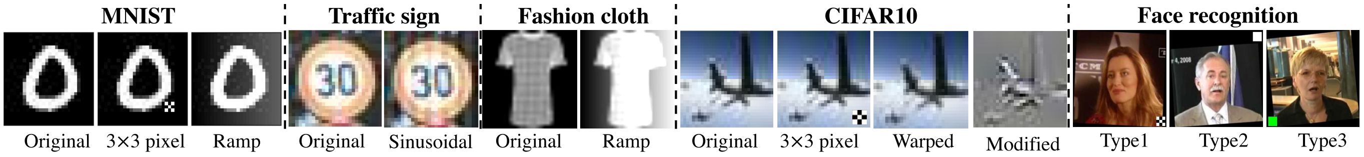


Fig. 4. Triggering signals. On CIFAR10, the 'Modified' image represents the difference between the original and warped image.

in Fig. 4, and modifying the sample labels into that of the target class.

- Corrupted-label attack, with ramp trigger (*ramp corrupted*): Eve performs a corrupted-label backdoor attack using a horizontal ramp signal [9] as a trigger (see Fig. 4). The ramp signal is defined as $\upsilon(i, j) = j\Delta/W$, $1 \leq i \leq H$, $1 \leq j \leq W$, where $H \times W$ is the size of the image. The parameter Δ controls the slope (and strength) of the ramp trigger signal, with a larger value leading to a more visible trigger (and to a more effective attack). We set $\Delta = 40$ in the experiments, that represents a good tradeoff between visibility and effectiveness of the attack [9].
- Clean-label attack, with 3 × 3 pixel trigger (3 × 3 *clean*): the 3 × 3 pixel trigger signal is utilised to perform a clean-label attack.

2) Traffic Signs: For the traffic sign classification task, we selected 16 different classes from the GTSRB dataset, namely, the most representative classes in the dataset, including 6 speed-limit, 3 prohibition, 3 danger, and 4 mandatory signs. Each class has 1200 colour images with size 28 × 28. The model architecture used for training is based on ResNet18 The feature representation is extracted from the 17-th layer, that is, before the FC layer, after an average pooling layer and ReLu activation. With regard to the attack, we considered the corrupted-label scenario. The triggering signal is a horizontal sinusoidal signal, defined as $\upsilon(i, j) = \Delta \sin(2\pi j f/W)$, $1 \leq i \leq H$, $1 \leq j \leq W$, where $H \times W$ is the size of input image. The parameters Δ and f are the amplitude and frequency of the sinusoidal function, which control the strength and frequency of the sinusoidal signal. In our experiment, we set $\Delta = 20$ and $f = 6$. The sinusoidal trigger is shown in Fig. 4. As before, for a given α, the network is trained on 16 poisoned datasets, each time considering different target classes.

3) Fashion Clothes: The Fashion-clothes dataset includes 10 classes of grey-level cloth images, each class consisting of 6000 images of size 28 × 28. The model architecture used for the classification is based on AlexNet with input size equal to 224 × 224. The representation used by the backdoor detector is extracted from the 5-th layer, at the output of the ReLu activation layer before the first FC layer. With regard to the attack, the poisoned samples are generated by performing the attack in a clean-label setting. A ramp trigger with $\Delta = 256$ is used to implement the attack, as shown in Fig. 4. Once again, for each choice of α, the backdoor attack is repeated 10 times, each time considering a different target class.

Furthermore, to prove the universality of the proposed method, we also run some experiments considering more complex tasks and realistic datasets, namely CIFAR10 classification and face recognition, and with different backdoor attacks relying on source-specific and sample-specific triggers. The setting used and the results achieved in these cases are reported in Section V-E.

For all the classification tasks, the benign validation dataset D_{val} is obtained by randomly selecting 100 samples from all the classes in the dataset.

C. Parameters Setting and State-of-the-Art Comparison

To implement CCA-UD, we have to set the following parameters: the reduced dimension d' for the clustering, the parameters of the DBSCAN algorithm, namely $minPts$ and ϵ, and finally the threshold θ used by the clustering poisoning detection module. In our experiments, we set $d' = 2$, $minPts = 20$ and $\epsilon = 0.8$. This is the setting that, according to our experiments, achieves the best performance with the minimum complexity for the clustering algorithm (being $d' = 2$).[10] The effect of these parameters on the result of clustering and detection performance is evaluated by the ablation study reported in the supplementary material (Section I).

We compared our method with the state-of-the-art methods mentioned in Section I-A, namely, AC [10], CI [13], SCAn [16] and Beatrix [17].

With regard to θ, as mentioned before, AC, CI, SCAn, Beatrix and CCA-UD involve the setting of a threshold for poisoning detection. Specifically, in AC the relative size of the class ratio is thresholded, while in CI we vary the prediction change rate between filtered and non-filtered samples. For SCAn, the threshold applies to the hypothesis testing statistics used to judge if a class is poisoned or not (the smallest size cluster obtained after the application of the LDA represents the set of poisoned data). Finally, for Beatrix, we varied the anomaly detection threshold

For a fair comparison, we set the threshold in the same way for all the methods. In particular, we set θ by fixing the average FPR on the validation dataset (consisting of benign samples). For each class D^i_{val}, we calculate the $FPR(D^i_{val})$ value, and its average counterpart is $\overline{FPR}(D_{val}) = \sum_i FPR(D^i_{val})/l$. We chose the threshold in such a way to minimise the distance from the target rate. Formally, by setting the target false positive rate to 0.05, the threshold θ^* is determined as:

$$\theta^* = \arg\min_{\theta} \left|0.05 - \overline{FPR}(D_{val})\right|. \tag{6}$$

The parameters of CCA-UD and their settings are summarized in Table II.

[10] Notably, the same setting works for all the cases, namely for all the tasks, architectures and attacks we have considered in our experiments.

TABLE II

SETTING OF DBSCAN PARAMETERS IN CCA-UD

Parameter	Setting	Meaning
d'	2	Dimension of the reduced space (UMAP)
ϵ	0.8	Neighbourhood radius (DBSCAN)
θ^*	Set via (6)	Threshold used in the PCD module
$minPts$	20	Minimal sample numbers in the neighbourhood of a core point (DBSCAN)

TABLE III

$\overline{AUC}$ OF THREE METHODS WITH THE VARIOUS ATTACKS

Method	**3×3 corrupted**	**Ramp corrupted**	**3×3 clean**
AC	0.728	0.733	0.785
CI	0.964	0.178	0.488
SCAn	0.848	0.868	0.984
Beatrix	0.991	0.908	0.990
CCA-UD	**0.994**	**0.996**	**0.981**

V. EXPERIMENTAL RESULTS

A. Threshold Setting

Our experiments reveal that, for AC and CI, the threshold determined via Eq. (6) does not lead to a good operating point when used on the test dataset (see Table IV for the MNIST case), and a threshold that works well in all cases can not be found for AC and CI. In particular, while for SCAn, Beatrix and CCA-UD, the threshold θ^* set on the validation dataset yields a similar $\overline{FPR}$ (around 0.05) in the BC_B, BC_P and PC cases, this is not true for AC and CI, for which $\overline{FPR}(BC_B)$, $\overline{FPR}(BC_P)$ and $\overline{FPR}(PC)$ are often smaller than 0.05, reaching 0 in many cases. This leads to a poor $\overline{TPR}(PC)$. In particular, with AC, when $\alpha > \theta^*$, both clusters are classified as benign, and then $\overline{TPR}_\alpha(PC) = \overline{FPR}_\alpha(PC) = 0$, even when the method would, in principle, be able to provide perfect discrimination ($\overline{AUC}_\alpha \approx 1$). The difficulty in setting the threshold for AC and CI is also evident from the plots in Fig. 5, that report the $\overline{FPR}$ and $\overline{TPR}$ values (that are averaged also on α), for different values of the threshold θ. From these plots, we clearly see that a threshold that works for all α can not be found for AC and CI.

Due to the difficulties encountered to set the detection threshold for AC and CI,[11] the results at θ^* for these methods are not reported in the other cases, for which we report only the $\overline{AUC}_\alpha$. Note that the possibility to set a unique threshold on a benign dataset that also works on poisoned datasets is very important for the practical applicability of a defence.

B. Results on MNIST

Performance is evaluated against the three types of backdoor attacks, namely, 3 × 3 *corrupted*, *ramp corrupted*, and 3 × 3 *clean*. In Fig. 5, the figures report the average performance of AC, CI, SCAn, Beatrix, and CCA-UD in the three cases. The values of $\overline{FPR}(BC_B)$, $\overline{FPR}(BC_P)$, $\overline{TPR}(PC)$ and $\overline{FPR}(PC)$ are reported for each method, as a function of the detection threshold θ. The behaviour of $\overline{FPR}(D_{val})$, which is utilised to determine the threshold θ^* (at 0.05 of $\overline{FPR}(D_{val})$), is also reported. The position of θ^* is indicated by a vertical dotted line.[12]

By observing the figure, we see that CCA-UD outperforms the other methods in all the settings. In the first setting, CCA-UD achieves $\overline{TPR}(PC)$ and $\overline{FPR}(PC)$ equal to 0.983 and 0.051 at the optimal threshold θ^*, with $\overline{FPR}(BC_B) = 0.051$ and $\overline{FPR}(BC_P) = 0.050$. The performance of SCAn and Beatrix is a bit worse than CCA-UD, while the performance of AC and CI at their optimal threshold is very poor. Similar results are achieved for the second and third settings. In particular, for the second attack, CCA-UD achieves $\overline{TPR}(PC)$ and $\overline{FPR}(PC)$ equal to (0.975, 0.050) at θ^*, and (0.966, 0.050) for the third one.

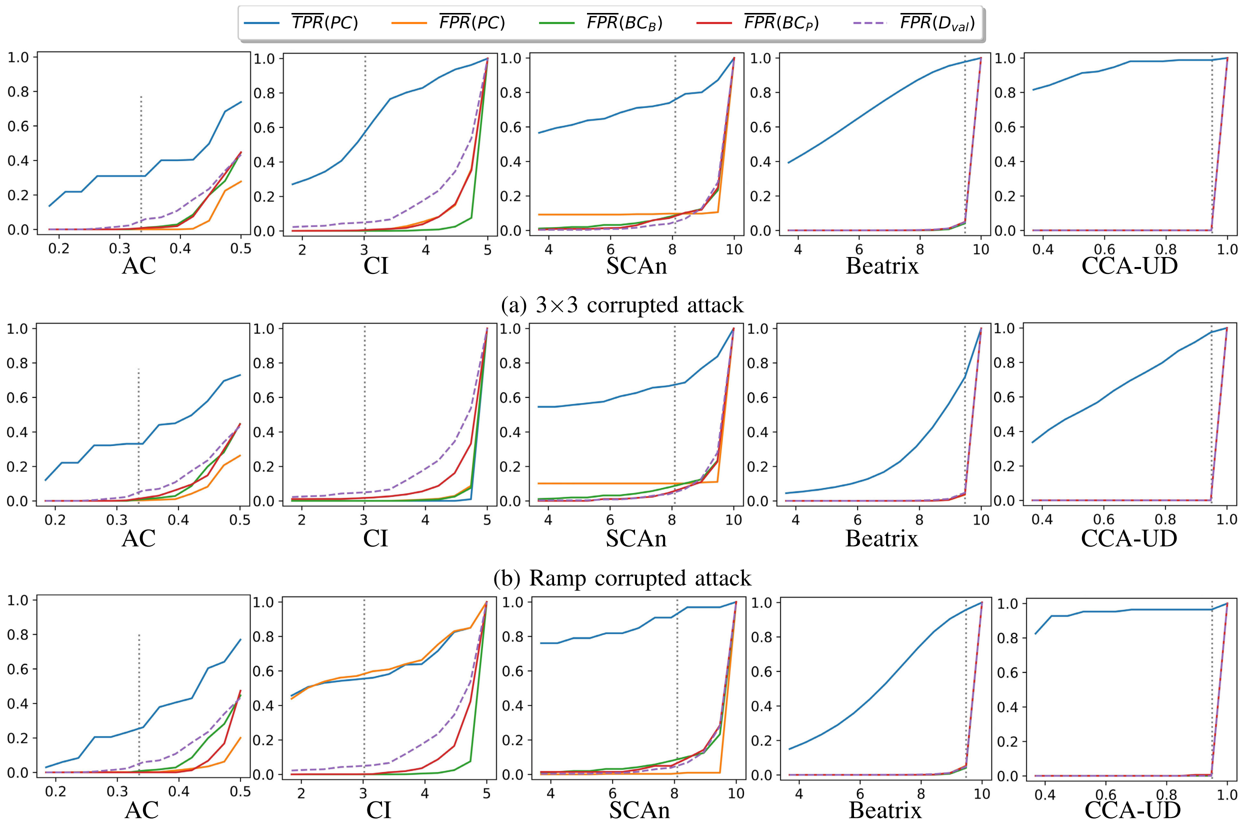
For a poisoned dataset, the $\overline{AUC}$ values obtained in the three settings are provided in Table III. From these results, we see that CI has good discriminating capability (with an AUC only slightly lower than CCA-UD) against the first attack, but fails to defend against the other two. This is an expected behaviour since CI does not work when the triggering signal is robust against the average filter, as the ramp signal considered in the second attack, or clean-label attacks in the last setting. SCAn and Beatrix instead have performance only slightly lower than our method.

Table IV shows the results obtained for different values of the poisoning ratio α for the three different attacks. The values of $\overline{FPR}_\alpha(PC)$ and $\overline{TPR}_\alpha(PC)$ have been obtained by letting $\theta = \theta^*$. For the clean-label case, due to the difficulty of developing a successful attack [5], [9], [30], the backdoor can be successfully injected in the model only when α is large enough and, in any case, a successful attack could not always be obtained in the 10 repetitions. The number of successfully attacked classes (cnt) with different poisoning ratios is reported in this case. Upon inspection of Table IV, we observe that:

- With regard to AC, the behaviour is similar under the three attack scenarios. Good results are achieved for intermediate values of α, namely in the $[0.2, 0.3]$ range. However, AC can not defend against backdoor attacks adopting a poisoning ratio smaller than 0.1. Moreover, AC does not work when $\alpha > 0.5$, in which case $\overline{AUC}_\alpha$ goes to zero, as benign samples are judged as poisoned and vice-versa (this is a consequence of the assumption made on the cluster size). Finally, by comparing Table IVa and IVc, we see that AC achieves better $\overline{AUC}_\alpha$ against the corrupted-label attack than in the clean-label case .
- With regard to CI, the detection performance achieved in the first attack scenario (3 × 3 corrupted) is good for all the values of α, (with the exception of the smallest α, for which $\overline{AUC}_\alpha = 0.876$), showing that CI can effectively defend against the backdoor attack in this setting, for every attack poisoning ratio. However, as expected, CI fails in the other settings.
- Regarding SCAn, when $\alpha < 0.5$ results are very good in all the settings, with the only notable exception of small α, in which case low $\overline{AUC}_\alpha$ and, in particular, low

[11] Note that the problem of threshold setting is not addressed in the original papers, since different thresholds are used in the various cases.

[12] We verified that the threshold on the false positive rate set on the validation dataset, also works on the test dataset, where we get the target FPR (0.05 in our case).



(a) 3×3 corrupted attack

(b) Ramp corrupted attack

(c) 3×3 clean attack

Fig. 5. Average performance of AC and CI, SCAn, Beatrix, and CCA-UD for different the threshold against the three types of backdoor attacks implemented in the case of MNIST classification. The position of θ^* is indicated by a vertical dotted line. In each figure, the x-axis represents the threshold value, while the y-axis reports the values of $\overline{FPR}(BC_B)$, $\overline{FPR}(BC_P)$, $\overline{TPR}(PC)$ and $\overline{FPR}(PC)$ and $\overline{FPR}(D_{val})$.

$\overline{TPR}_\alpha(PC)$ values are achieved. Similarly to AC, when $\alpha > 0.5$, $\overline{AUC}_\alpha$ goes to zero.

- With regard to Beatrix, results are good especially in the first and third setting, a little bit worse in the second setting. Also in this case, the values of $\overline{TPR}_\alpha(PC)$ and $\overline{AUC}_\alpha$ are lower when α is small.
- CCA-UD provides good results in all settings and for every value of α, with a perfect or nearly perfect $\overline{AUC}_\alpha$ in most of the cases. Moreover, a very good $\overline{TPR}_\alpha(PC)$ is obtained, always larger (often much larger) than 0.95 with only very few exceptions in the corrupted attack settings when α is very low, and a $\overline{FPR}_\alpha(PC)$ around 0.05.

Overall, the tables confirm the universality of CCA-UD that works very well in all the setting, regardless of the specific attack and regardless of the value of α, thus confirming that the strategy used to detect poisoned clusters exploits a general misclassification behaviour.

C. Results on Traffic Signs

Fig. 6 shows the average performance of AC, CI, SCAn, Beatrix, and CCA-UD on traffic signs. Similar considerations to the MNIST case can be made. CCA-UD achieves very good average performance at the operating point given by θ^*, where $\overline{TPR}(PC)$ and $\overline{FPR}(PC)$ are (0.965, 0.058) (with $\overline{FPR}(BC_B) = \overline{FPR}(BC_B) \approx 0.08$). As before, for AC and CI a threshold that works well on the average can not be found. In the case of a poisoned dataset, the average AUC of the detection $\overline{AUC}$ is equal to 0.897, 0.958, 0.924, 0.965, 0.993 for AC, CI, SCAn, Beatrix and CCA-UD, respectively. We observe that also CI gets a good $\overline{AUC}_\alpha$. In fact, given that the size of the input image is 28 × 28, the trigger, namely the sinusoidal signal can be effectively removed by a 5×5 average filter.

The results obtained for various α are reported in Table V. As it can be seen, CCA-UD achieves the best performance in terms of $\overline{AUC}_\alpha$, and $\overline{TPR}_\alpha(PC)$ and $\overline{FPR}_\alpha(PC)$ measured at $\theta = \theta^*$, in all the cases. With regard to the other methods, as observed before, while CI and Beatrix are relatively insensitive to α, the performance of AC and SCAn drop when $\alpha < 0.1$ or $\alpha > 0.5$. Also in this case, CCA-UD is the best-performing method outperforming Beatrix (second-best), with a gain of about 0.03 in $\overline{AUC}_\alpha$ for every setting of α, and up to 0.12 in $\overline{TPR}_\alpha$.

D. Results on Fashion Clothes

Fig. 7 reports the results obtained on the fashion clothes task. We did not run the SCAn method in this case. In fact, as already pointed in [17], SCAn is highly time-consuming when the feature dimension is large. With this network $d = 9216$ (with MNIST and traffic sign d is lower than 512), SCAn took more than 7 days to find out the poisoned samples from the training dataset, running on a computer with Intel(R) Core(TM) i7-8700 CPU@3.20GHz.

TABLE IV

PERFORMANCE FOR VARIOUS POISONING RATIOS α, AGAINST THE THREE TYPES OF BACKDOOR ATTACKS FOR MNIST CLASSIFICATION. $\overline{FPR}_\alpha(PC)$ AND $\overline{TPR}_\alpha(PC)$ (INDICATED AS $\overline{FPR}_\alpha$ AND $\overline{TPR}_\alpha$ IN THE TABLE) VALUES ARE COMPUTED WITH $\theta = \theta^*$. IN SUB-TABLE (C) CNT INDICATES THE NUMBER OF SUCCESSFUL ATTACKS IN 10 REPETITIONS

	AC			**CI**			**SCAn**			**Beatrix**			**CCA-UD**		
α	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$
0.025	0.000	0.000	0.563	0.324	0.022	0.876	0.207	0.061	0.810	0.868	0.057	0.934	0.908	0.051	0.949
0.050	0.099	0.000	0.628	0.581	0.001	0.977	0.696	0.000	0.987	0.967	0.048	0.992	0.989	0.050	0.994
0.096	0.395	0.000	0.757	0.654	0.000	0.996	0.992	0.000	0.996	0.995	0.046	0.999	0.999	0.050	0.999
0.134	0.792	0.000	0.958	0.559	0.002	0.990	0.994	0.000	0.997	0.998	0.060	1.000	0.999	0.050	1.000
0.186	0.994	0.000	0.997	0.577	0.001	0.985	0.996	0.000	0.998	0.999	0.054	1.000	1.000	0.050	1.000
0.258	0.993	0.000	0.997	0.540	0.070	0.961	0.995	0.000	0.998	0.999	0.048	1.000	1.000	0.050	1.000
0.359	0.000	0.000	0.998	0.571	0.005	0.964	0.996	0.000	0.998	0.999	0.053	1.000	1.000	0.050	1.000
0.550	0.000	0.000	0.001	0.829	0.000	0.953	0.002	1.000	0.001	0.996	0.043	0.999	1.000	0.050	1.000

(a) 3×3 corrupted

	AC			**CI**			**SCAn**			**Beatrix**			**CCA-UD**		
α	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$
0.035	0.050	0.024	0.593	0.000	0.008	0.407	0.222	0.020	0.948	0.423	0.037	0.777	0.871	0.050	0.966
0.050	0.090	0.028	0.593	0.000	0.000	0.119	0.300	0.000	0.999	0.552	0.036	0.851	0.914	0.050	0.998
0.096	0.400	0.000	0.786	0.000	0.000	0.216	0.773	0.000	0.999	0.739	0.037	0.926	0.989	0.050	0.998
0.134	0.798	0.001	0.962	0.000	0.000	0.142	0.999	0.000	0.999	0.732	0.041	0.910	0.999	0.050	0.998
0.186	0.992	0.003	0.995	0.000	0.000	0.179	1.000	0.000	1.000	0.781	0.042	0.934	1.000	0.050	1.000
0.258	0.999	0.000	0.999	0.000	0.000	0.088	1.000	0.000	1.000	0.774	0.035	0.941	1.000	0.050	1.000
0.359	0.000	0.000	0.999	0.000	0.000	0.144	1.000	0.000	1.000	0.834	0.048	0.953	1.000	0.050	1.000
0.550	0.000	0.000	0.002	0.000	0.000	0.135	0.006	1.000	0.000	0.873	0.033	0.979	1.000	0.050	1.000

(b) Ramp corrupted

		AC			**CI**			**SCAn**			**Beatrix**			**CCA-UD**		
α	cnt	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$
0.069	3	0.000	0.000	0.533	0.667	0.667	0.296	0.758	0.000	0.977	0.937	0.032	0.987	0.952	0.050	0.972
0.096	3	0.000	0.000	0.528	0.333	0.333	0.595	0.959	0.000	0.980	0.919	0.067	0.973	0.951	0.050	0.972
0.134	3	0.000	0.000	0.610	0.667	0.667	0.539	0.976	0.000	0.988	0.961	0.052	0.988	0.975	0.050	0.987
0.186	5	0.384	0.003	0.746	0.600	0.600	0.471	0.970	0.000	0.985	0.973	0.052	0.993	0.982	0.050	0.991
0.258	5	0.929	0.011	0.959	0.601	0.644	0.516	0.988	0.000	0.994	0.974	0.057	0.996	0.994	0.051	0.996
0.359	5	0.315	0.000	0.975	0.206	0.213	0.437	0.959	0.000	0.979	0.991	0.069	0.997	0.993	0.050	0.996
0.450	5	0.000	0.000	0.969	0.729	0.786	0.554	0.978	0.000	0.989	0.986	0.061	0.996	0.997	0.050	0.998

(c) 3×3 clean

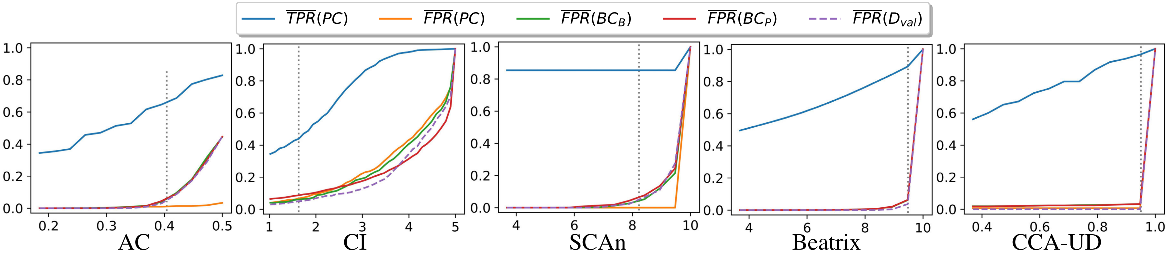


Fig. 6. Average performance of AC, CI, SCAn, Beatrix, and CCA-UD for different values of θ for the traffic signs task. The vertical dotted line indicates the position of θ^* for the various methods. line. In each figure, the x-axis represents the threshold value, while the y-axis reports the values of $\overline{FPR}(BC_B)$, $\overline{FPR}(BC_P)$, $\overline{TPR}(PC)$ and $\overline{FPR}(PC)$ and $\overline{FPR}(D_{val})$.

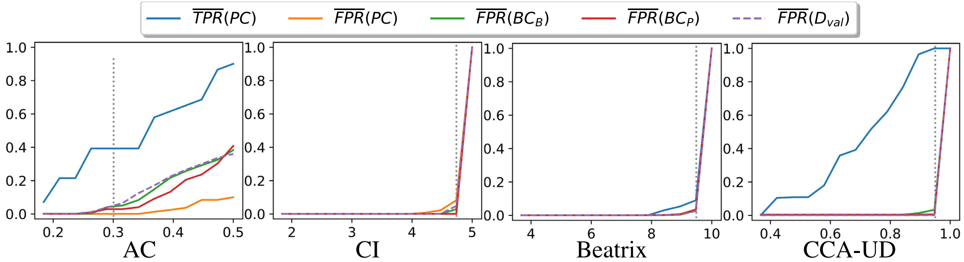


Fig. 7. Average performance of AC, CI, Beatrix, and CCA-UD for different values of θ for the fashion clothes task. The vertical dotted line indicates the position of θ^* for the various methods. In each figure, the x-axis indicates threshold values, while the y-axis reports the values of $\overline{FPR}(BC_B)$, $\overline{FPR}(BC_P)$, $\overline{TPR}(PC)$ and $\overline{FPR}(PC)$ and $\overline{FPR}(D_{val})$.

Once again, the performance of CCA-UD is largely superior to those achieved by other works. In particular, by looking at Fig. 7, CCA-UD achieves $\overline{TPR}(PC)$ and $\overline{FPR}(PC)$ equal to (1.000, 0.053), with $\overline{FPR}(BC_B) = \overline{FPR}(BC_P) \approx 0.05$. Regarding the AUC scores, $\overline{AUC}$ of AC, CI, Beatrix, and CCA-UD are 0.900, 0.106, 0.519 and 0.997 respectively.

TABLE V

PERFORMANCEFOR VARIOUS POISONING RATIOS FOR THE TRAFFIC SIGNS TASK. $\overline{FPR}_\alpha(PC)$ AND $\overline{TPR}_\alpha(PC)$ (INDICATED AS $\overline{FPR}_\alpha$ AND $\overline{TPR}_\alpha$ IN THE TABLE) VALUES ARE COMPUTED WITH $\theta = \theta^*$

		AC	**CI**	**SCAn**			**Beatrix**			**CCA-UD**		
α	cnt	$\overline{AUC}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$
0.050	9	0.793	0.923	0.728	0.000	0.864	0.929	0.075	0.973	0.946	0.061	0.983
0.096	9	0.850	0.928	0.844	0.000	0.922	0.910	0.079	0.973	0.998	0.059	0.991
0.134	9	0.949	0.959	0.884	0.000	0.942	0.878	0.071	0.958	0.998	0.057	0.992
0.186	10	0.958	0.965	0.867	0.000	0.934	0.873	0.071	0.961	0.999	0.056	0.993
0.359	13	0.946	0.965	0.925	0.000	0.963	0.897	0.079	0.960	0.985	0.054	0.996
0.450	14	0.917	0.965	0.908	0.000	0.954	0.915	0.070	0.963	0.980	0.055	0.994
0.550	15	0.869	0.996	0.782	0.000	0.891	0.934	0.085	0.971	0.999	0.051	0.999

TABLE VI

PERFORMANCEFOR VARIOUS POISONING RATIOS FOR THE FASHION CLOTHES TASK. $\overline{FPR}_\alpha(PC)$ AND $\overline{TPR}_\alpha(PC)$ (INDICATED AS $\overline{FPR}_\alpha$ AND $\overline{TPR}_\alpha$) VALUES ARE COMPUTED WITH $\theta = \theta^*$

		AC	**CI**	**Beatrix**			**CCA-UD**		
α	cnt	$\overline{AUC}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$	$\overline{TPR}_\alpha$	$\overline{FPR}_\alpha$	$\overline{AUC}_\alpha$
0.069	3	0.618	0.056	0.113	0.059	0.546	1.000	0.052	0.998
0.096	3	0.513	0.341	0.025	0.047	0.456	1.000	0.056	0.995
0.134	3	0.940	0.087	0.025	0.074	0.508	1.000	0.053	0.998
0.186	4	1.000	0.037	0.032	0.059	0.425	1.000	0.055	0.998
0.258	5	1.000	0.083	0.026	0.061	0.413	1.000	0.057	0.996
0.359	5	1.000	0.015	0.300	0.060	0.697	1.000	0.052	0.998
0.450	5	1.000	0.174	0.207	0.059	0.591	1.000	0.050	1.000

Since the attack is carried out in a clean-label modality, the poor performance of CI is expected. The bad performance of Beatrix in this case, instead, is likely due to the curse of dimensionality, since distinguishing between poisoned samples and benign ones by relying on inner products of feature vectors is not possible when the feature dimension is very large (as it is the case here, with a dimension larger than 9000). The results for various α are reported in Table VI and confirm the good behaviour of CCA-UD, which provides very good performance in all the cases, always outperforming the other methods.

E. Other Datasets, Architectures and Attacks

Below we report the additional experiments we carried out with different and more complex tasks and attacks, to prove the generality/universality of the proposed method.

1) CIFAR10 Classification: CIFAR10 dataset consists of a total of 60000 images of size 32×32 belonging to 10 classes, split into two parts (50000 for training and 10000 for testing). The model architecture is based on VGG19 , and the feature representation is extracted from the final convolutional (16th) layer after the pooling layer and flatten operation. The following types of corrupted-label backdoor attacks are considered:

- Corrupted-label attack, with 3×3 pixel trigger, see Fig. 4. Specifically, the attacker chooses samples from non-target classes, adds the trigger over the samples, and finally modifies the labels to the target class.
- Source-specific attack, with 3×3 pixel trigger, see Fig. 4. The attacker poisons samples from a specific source class and modifies their labels into that of the target class. At test time, only poisoned samples from this class can lead to misclassification.
- Sample-specific attack. The attack is carried out considering the warping-based trigger described in [42], see Fig. 4. The poisoned images are warped with fixed parameters and the labels modified into that of the target class. To facilitate the network to learn the specific backdoor, the attacker also injects into the training dataset noise samples, namely images warped randomly, for which the labels are not corrupted.

In this case, to speed up the experiments, we run our tests by randomly choosing the target class, instead of repeating the experiments for every possible choice of the target. Experiments are carried out considering 5 different poisoning ratios, ranging from 0.096 to 0.45.

The $\overline{AUC}_\alpha$ obtained for the three types of attack are shown in Fig. 8a, Fig. 8b, and Fig. 8c, respectively. We verified that SCAn, Beatrix, and CCA-UD[13] can achieve average FPR, evaluated on the benign class, as $0.066, 0.052$, and 0.003, respectively.

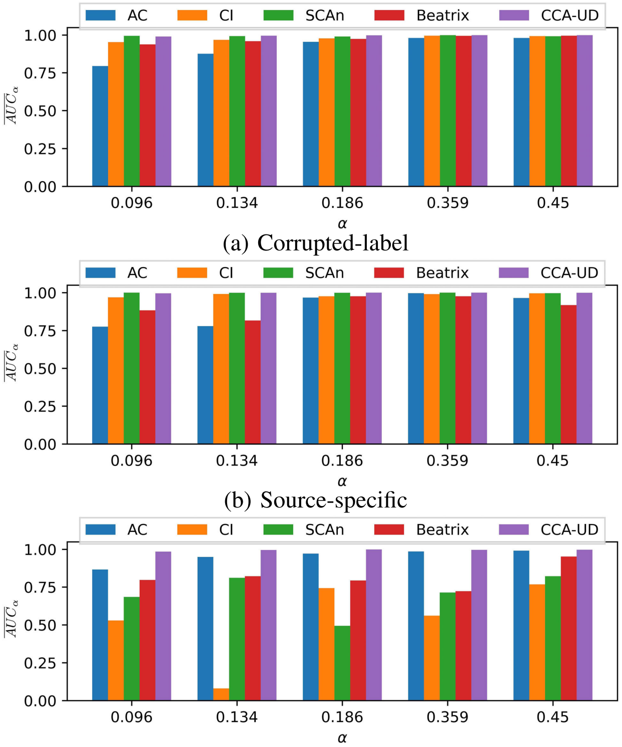
For the corrupted-label attack in Fig. 8a, we can observe that CI, SCAn, Beatrix and CCA-UD can achieve good performance for different poisoning ratios with $\overline{AUC}_\alpha \approx 1$. However, AC performance degrades to 0.77 when the poisoning ratio decreases.

In the case of the source-specific attack shown in Fig. 8b, all methods work very well when $\alpha \geq 0.186$. AC and Beatrix are the methods showing the worse performance (with an $\overline{AUC}_\alpha \approx 0.75$) when α is small ($\alpha \leq 0.134$ in the plots).

By choosing the threshold as explained in Section IV-C, we were able to find a threshold θ^* that works in all cases, also for this task. Specifically, by choosing the threshold in this way (set to 9.74, 9.38, 0.98) we get $\overline{TPR}(PC)$ and $\overline{FPR}(PC)$ equal to (0.775, 0.001), (0.671, 0.078), (0.996, 0.002) for different α's and three different tasks. .

From the results obtained against the sample-specific attack shown in Fig. 8c, we can observe that CCA-UD achieves the best performance with $\overline{AUC}_\alpha \approx 1$ in all the cases. With regard to the other methods, we can observe the following: 1) AC achieves good results when α is large (being always smaller than 0.5), while - as before - performance drops when α becomes very small; 2) CI is not very effective, since the average filter cannot remove the warping-based trigger; 3) SCAn's bad performance was expected since, as observed in [17], this method can not work on sample-specific attacks Beatrix can indeed improve the performance of SCAn in this case, however, performance is good only when α is large.

[13] For AC and CI, it is hard to find a fixed threshold, as discussed in Section V-A, so we only compare CCA-UD with SCAn and Beatrix.



(a) Corrupted-label

(b) Source-specific

(c) Sample-specific

Fig. 8. Performance ($\overline{AUC}_\alpha$) of AC, CI, SCAn, Beatrix and CCA-UD for various poisoning ratio α for CIFAR10 classification, against the three types of backdoor attacks. .

2) Face Recognition Task: We also run some experiments considering a very different task, namely, face recognition, with larger-size images. For this task, features tend to have a larger intra-class variability compared to the other tasks. To prove this, we visualise the feature distribution for this task in Section III of the supplementary material and compare it with CIFAR10.

For these experiments, the 12 most populated classes in YoutubeFace dataset [43] were selected, with more than 2600 images per class. The dataset is split in training and testing in proportions 9:1. The image size is 315 × 315. Classification is performed considering an Inception-Resnet-v1 architecture [44]. The feature representation for the clustering analysis are extracted before the first FC layer (2nd last layer).

With regard to the attack, we considered the corrupted label case, with a 30 × 30 pixel trigger, see Type1 trigger in Fig. 4. The same triggering signal used before was considered, enlarged by a factor of 10 (being the size of the images approximately 10 times larger than in the previous cases, this triggering signal has approximately the same relative size as before).

As for the case of CIFAR10, we run tests for a randomly chosen target class. Experiments were carried out by considering 5 different poisoning ratios, i.e., 0.05, 0.134, 0.359, 0.45, 0.55.

Fig. 9 shows the $\overline{AUC}_\alpha$ values. Performance is good for most of the methods. CCA-UD and Beatrix are the best-performing methods, with an almost perfect $\overline{AUC}_\alpha$ in all the settings. With regard to the other methods, a behaviour similar to the previous cases can be observed. In particular, AC and SCAn do not work when $\alpha = 0.55$, and their performance drops (especially for SCAn) when α is very small (cluster imbalance issue). CI is the method that has worse performance on average, however, some discrimination capability can be observed also for this method. In fact, even if the average filter kernel is 5×5 (and the filter can not completely remove the trigger), the triggering signal is impaired by the filter and the activation of the backdoor inhibited.

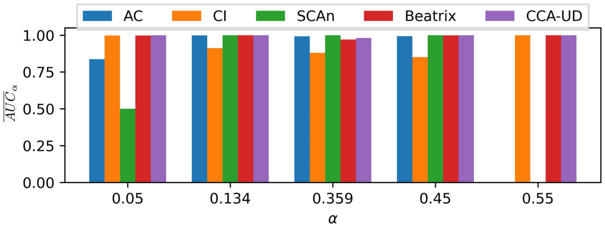


Fig. 9. Performance ($\overline{AUC}_\alpha$) of AC, CI, SCAn, Beatrix and CCA-UD for various poisoning ratio α for face recognition (YouTubeFace), against the 30 × 30 pixel trigger attacks.

For FPR, we evaluated the performance of SCAn, Beatrix and CCA-UD on the benign class and got average values equal to 0.086, 0.053, and 0.002. Given the thresholds (8.46, 9.65, and 0.95 for SCAn, Beatrix and CCA-UD) determined as in Section V-A, the average $\overline{TPR}(PC)$ and $\overline{FPR}(PC)$ are equal to (0.800, 0.100), (0.961, 0.053), (0.988, 0.000), respectively.

In summary, these results confirm that CCA-UD is effective also when there is large intra-variability in the feature representation of the various classes and then the samples in benign classes are split into distinct clusters (as shown in Section III of supplementary material). Being these clusters benign, as expected, the clusters' centroids do not activate the misclassification behaviour CCA-UD looks for.

For the face recognition task, we also run an experiment considering a multiple triggers attack, to evaluate the effectiveness of CCA-UD also in this scenario. In a multiple triggers attack, several triggers are used to poison the samples, to induce more than one malicious behaviour. Specifically, in our experiments, the attacker chooses three types of 30 × 30 triggers to poison three different classes, as shown in Fig. 4, where the three triggers use different patterns and are placed in different locations. At test time, the presence of the triggering signal inside the sample will lead to a misclassification in favour of the corresponding target class.

Our experiments confirm that CCA-UD can achieve very good performance, with an average $\overline{AUC} \approx 0.99$, for all the target classes. At the optimum threshold $\theta^* = 0.95$ (this threshold is the same as for the previous experiment as it was set on benign data), we get an error probability averaged over the three classes equal to $\overline{TPR}(PC) = 0.997$ and $\overline{FPR}(PC) = 0.006$.

On the same experiments SCAn and Beatrix achieved $\overline{AUC} = 0.833$ and $\overline{AUC} = 0.999$ respectively, and identified the poisoned samples with $\overline{TPR}(PC)$ and $\overline{FPR}(PC)$ equal to (0.667, 0.001) and (0.998, 0.064), respectively.

VI. Concluding Remarks

We have proposed a universal backdoor detection method, called CCA-UD, to reveal the possible presence of a backdoor inside a model and identify the poisoned samples by analysing the training dataset. CCA-UD relies on DBSCAN clustering and on a new strategy for the detection of poisoned clusters based on the analysis of clusters' centroids, that exploits a general behaviour of backdoored models. The capability of the centroids' features to cause misclassification of benign samples is exploited to decide whether a cluster is poisoned or not. We evaluated the effectiveness of CCA-UD on a great variety of classification tasks and architectures, and attack scenarios. The results confirm that CCA-UD can work well regardless of the corruption strategy (corrupted- or clean-label), the poisoning ratio (that can either be very small or very large), and the type of trigger used by the attacker. In particular, in our experiments, we considered a wide variety of triggers, from fixed triggering signals (local and global pattern) to source-specific and sample-specific triggers. Furthermore, we proved that the performance achieved by CCA-UD is always superior or comparable to those achieved by the existing methods, when these methods are applied in a setting that meets their operating requirements.

Future work will be devoted to the investigation of the capability of CCA-UD to defend against backdoor attacks in application scenarios beyond image classification, The effectiveness of the method against backdoor attacks carried out against modern architectures (e.g. vision transformers) or different deep neural network models, e.g. generative models [45], sequential models [46], and recurrent neural networks in particular, is also worth investigation.

References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. ICLR*, 2015, pp. 1–11.

[2] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proc. 29th Int. Conf. Mach. Learn.*, 2012, pp. 1–8.

[3] W. Guo, B. Tondi, and M. Barni, "A master key backdoor for universal impersonation attack against DNN-based face verification," *Pattern Recognit. Lett.*, vol. 144, pp. 61–67, Apr. 2021.

[4] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv:1712.05526*.

[5] W. Guo, B. Tondi, and M. Barni, "A temporal chrominance trigger for clean-label backdoor attack against anti-spoof rebroadcast detection," *IEEE Trans. Depend. Sec. Comput.*, early access, Jan. 2, 2023, doi: 10.1109/TDSC.2022.3233519.

[6] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying vulnerabilities in the machine learning model supply chain," 2017, *arXiv:1708.06733*.

[7] W. Guo, B. Tondi, and M. Barni, "An overview of backdoor attacks against deep neural networks and possible defences," *IEEE Open J. Signal Process.*, vol. 3, pp. 261–287, 2022.

[8] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," 2019, *arXiv:1912.02771*.

[9] M. Barni, K. Kallas, and B. Tondi, "A new backdoor attack in CNNS by training set corruption without label poisoning," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 101–105.

[10] B. Chen et al., "Detecting backdoor attacks on deep neural networks by activation clustering," in *Proc. AAAI*, vol. 2301, 2019, pp. 1–8.

[11] J. Yadav and M. Sharma, "A review of K-mean algorithm," *Int. J. Eng. Trends Technol.*, vol. 4, no. 7, pp. 2972–2976, Jul. 2013.

[12] *Overview of Clustering Methods*. [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html

[13] Z. Xiang, D. J. Miller, and G. Kesidis, "A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense," in *Proc. IEEE 29th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Oct. 2019, pp. 1–6.

[14] S. R. Bond, A. Hoeffler, and J. R. Temple, "GMM estimation of empirical growth models," *J. SSRN*, vol. 1, pp. 1–37, Nov. 2001.

[15] A. A. Neath and J. E. Cavanaugh, "The Bayesian information criterion: Background, derivation, and applications," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 4, no. 2, pp. 199–203, Feb. 2012.

[16] D. Tang, X. Wang, H. Tang, and K. Zhang, "Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 1541–1558.

[17] W. Ma, D. Wang, R. Sun, M. Xue, S. Wen, and Y. Xiang, "The 'Beatrix' resurrections: Robust backdoor detection via Gram matrices," in *Proc. Netw. Distrib. Syst. Secur. (NDSS)*, 2023, pp. 1–18.

[18] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8011–8021.

[19] J. Hayase, W. Kong, R. Somani, and S. Oh, "SPECTRE: Defending against backdoor attacks using robust statistics," 2021, *arXiv:2104.11315*.

[20] H. Abdi, "Singular value decomposition (SVD) and generalized singular value decomposition," in *Encyclopedia of Measurement and Statistics*. Newbury Park, CA, USA: Sage, 2007, pp. 907–912.

[21] S. Shan, A. N. Bhagoji, H. Zheng, and B. Y. Zhao, "Poison forensics: Traceback of data poisoning attacks in neural networks," in *Proc. 31st USENIX Secur. Symp.*, K. R. B. Butler and K. Thomas, Eds. 2022, pp. 3575–3592.

[22] N. Peri et al., "Deep *k*-NN defense against clean-label data poisoning attacks," in *Proc. ECCV*, vol. 12535, 2020, pp. 55–70.

[23] Y. Gao et al., "On the effectiveness of adversarial training against backdoor attacks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 14, 2023, doi: 10.1109/TNNLS.2023.3281872.

[24] M. Weber, X. Xu, B. Karlaš, C. Zhang, and B. Li, "RAB: Provable robustness against backdoor attacks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2023, pp. 1311–1328, doi: 10.1109/sp46215.2023.10179451.

[25] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1310–1320.

[26] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, 2018, pp. 1–23.

[27] Z. Zhang, Q. Liu, Z. Wang, Z. Lu, and Q. Hu, "Backdoor defense via deconfounded representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Vancouver, BC, Canada, Jun. 2023, pp. 12228–12238.

[28] X. Gong et al., "Redeem myself: Purifying backdoors in deep learning models using self attention distillation," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2023, pp. 755–772.

[29] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, pp. 226–231.

[30] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, and Y.-G. Jiang, "Clean-label backdoor attacks on video recognition models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14431–14440.

[31] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[32] L. Rokach and O. Maimon, "Clustering methods," in *Data Mining and Knowledge Discovery Handbook*. Boston, MA, USA: Springer, 2005, pp. 321–352.

[33] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," *ACM SIGMOD. Rec.*, vol. 28, no. 2, pp. 49–60, Jun. 1999.

[34] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl. Discovery Data*, vol. 10, no. 1, pp. 1–51, Jul. 2015.

[35] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2018, *arXiv:1802.03426*.

[36] N. M. Jebreel, J. Domingo-Ferrer, and Y. Li, "Defending against backdoor attacks by layer-wise feature analysis," 2023, *arXiv:2302.12758*.

[37] M. Köppen, "The curse of dimensionality," in *Proc. 5th Online World Conf. Soft Comput. Ind. Appl. (WSC5)*, vol. 1, Sep. 2000, pp. 4–8.

[38] J. Duchon, "Splines minimizing rotation-invariant semi-norms in Sobolev spaces," in *Constructive Theory of Functions of Several Variables*, vol. 571, W. Schempp and K. Zeller, Eds. Berlin, Germany: Springer-Verlag, 1977, pp. 85–100.

[39] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," in *Proc. IEEE 7th Eur. Symp. Secur. Privacy (EuroS&P)*, Jun. 2022, pp. 703–718.

[40] M. Xue, C. He, J. Wang, and W. Liu, "One-to-N & N-to-One: Two advanced backdoor attacks against deep learning models," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 3, pp. 1562–1578, May 2022.

[41] PyTorch. *4-Layer DNN Model*. [Online]. Available: https://github.com/pytorch/examples/blob/main/mnist/main.py#L11

[42] T. A. Nguyen and A. T. Tran, "WaNet—Imperceptible warping-based backdoor attack," in *Proc. ICLR*, 2021, pp. 1–16.

[43] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *Proc. CVPR*, Jun. 2011, pp. 529–534.

[44] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 4278–4284.

[45] S. Chou, P. Chen, and T. Ho, "How to backdoor diffusion models?" in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Vancouver, BC, Canada, Jun. 2023, pp. 4015–4024.

[46] R. Nallapati, F. Zhai, and B. Zhou, "SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents," in *Proc. 31st AAAI Conf. Artif. Intell.*, vol. 31, no. 1, 2017, pp. 1–7.

Wei Guo received the B.Sc. degree from the Department of Mathematics and Computational Science, GUET, in 2015, and the M.Eng. degree from the Department of Computer and Information Security, Guilin University of Electronic Technology (GUET), in 2018, with a thesis about "Applied Cryptography in IoT environment". He is a Ph.D. candidate with the Department of Information Engineering and Mathematics of University of Siena (UNISI), Siena, Italy. He is currently doing research on security concerns in deep neural networks under the supervision of Prof. Mauro Barni. He is a member of Visual Information Privacy and Protection Group (VIPP).

Benedetta Tondi (Member, IEEE) received the master's degree (cum laude) in electronics and communications engineering and the Ph.D. degree in information engineering and mathematical sciences from the University of Siena, Siena, Italy, in 2012 and 2016, respectively, with a focus on the theoretical foundations of adversarial detection and applications to multimedia forensics, in the area of multimedia security.

From October 2014 to February 2015, she was a Visiting Student with the Signal Processing in Communications Group, University of Vigo, Vigo, Spain, involved in the study of techniques to reveal attacks in watermarking systems. She is currently an Assistant Professor with the Department of Information Engineering and Mathematics, University of Siena. She has been an Assistant for the course of information theory and coding and multimedia security. She is a member of the Visual Information Processing and Protection Group led by Prof. Mauro Barni. Recently, she has been working on machine learning and deep learning applications for digital forensics and counter-forensics, and on the security of machine learning techniques. Her stay was funded by a Spanish National Project on Multimedia Security. Her research interests include the application of information-theoretic methods and game theory concepts to forensics and counter-forensics analysis and more in general to multimedia security, and adversarial signal processing.

Dr. Tondi has been a member of the Information Forensics and Security Technical Committee, IEEE Signal Processing Society, since January 2019. She is a part of the IEEE Young Professionals and IEEE Signal Processing Society, and a member of the National Inter-University Consortium for Telecommunications (CNIT).

Mauro Barni (Fellow, IEEE) received the degree in electronic engineering from the University of Florence in 1991 and the Ph.D. degree in informatics and telecommunications in October 1995.

He has carried out his research activity for more than 20 years, first with the Department of Electronics and Telecommunication, University of Florence, and then with the Department of Information Engineering and Mathematics, University of Siena, where he is currently working as a Full Professor. His activity focuses on digital image processing and information security, with particular reference to the application of image processing techniques to copyright protection (digital watermarking) and authentication of multimedia (multimedia forensics). He has been studying the possibility of processing signals that have been previously encrypted without decrypting them (signal processing in the encrypted domain-s.p.e.d.). Then, he has been working on theoretical and practical aspects of adversarial signal processing and adversarial machine learning. His papers on digital watermarking have significantly contributed to the development of such a theory in the last decade as it is demonstrated by the large number of citations some of these papers have received. He is the author/coauthor of about 350 papers published in international journals and conference proceedings. He holds four patents in the field of digital watermarking and one patent dealing with anti-counterfeiting technology. His overall citation record amounts to an H-index of 63 according to Google Scholar search engine. He is also the coauthor of the book *Watermarking Systems Engineering: Enabling Digital Assets Security and Other Applications* (Dekker Inc., February 2004). He is an Editor of the book *Document and Image Compression* (CRC-Press, 2006).

Dr. Barni was a recipient of the *IEEE Signal Processing Magazine* Best Column Award in 2008, the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING Best Paper Award in 2010, and the Individual Technical Achievement Award of EURASIP in 2016. He was the Chairperson of the IEEE Multimedia Signal Processing Workshop held in Siena, in 2004, and the IVth edition of the International Workshop on Digital Watermarking. He was also the Technical Program Co-Chair of ICASSP 2014 and the Technical Program Chairperson of the 2005th edition of the Information Hiding Workshop, the VIIIth edition of the International Workshop on Digital Watermarking, and the Vth edition of the IEEE Workshop on Information Forensics and Security (WIFS 2013).

Open Access funding provided by 'Università degli Studi di Siena' within the CRUI CARE Agreement