

2DSig-Detect: a semi-supervised framework for anomaly detection on image data using 2D-signatures

Xinheng Xie^a, Kureha Yamaguchi^{b,c}, Margaux Leblanc^{b,c}, Simon Malzard^{b,c}, Varun Chhabra^{b,c}, Victoria Nockles^b, Yue Wu^{a,d}

^a*Department of Mathematics and Statistics at University of Strathclyde Glasgow G1 1XH UK.*

^b*Defence AI Research (DARe) The Alan Turing Institute London NW1 2DB UK.*

^c*equal contribution and VN is the PI of The Alan Turing Institute DARe team.*

^d*corresponding author e-mail: yue.wu@strath.ac.uk*

Abstract

The rapid and widespread deployment of machine learning within critical systems raises serious questions about their security against adversarial attacks. Models performing image-related tasks, are vulnerable to integrity violations causing misclassifications that do not compromise normal system operation, but rather, produce undesirable outcomes for targeted inputs. This paper introduces a novel technique for anomaly detection in images called *2DSig-Detect*, which is a 2D-signature-embedded semi-supervised framework rooted in rough path theory. We demonstrate that 2D-signatures can be applied to detect both adversarial examples and backdoored data, mitigating against test-time and training-time integrity attacks. Our results demonstrate both the efficacy and superior computational efficiency of our 2D-signature method for detecting adversarial manipulations to the training and test data.

Keywords: 2D-signature, rough path, evasion attacks, backdoor attacks, anomaly detection, semi-supervised learning

1. Introduction

In recent years, AI has advanced significantly, especially in tasks such as image and object recognition as well as image generation [1][2][3]. However, these models are quite vulnerable; even tiny modifications can disrupt their functionality. At the data level, the two major threats are data poisoning [4][5] and evasion attacks [6][7]. Data poisoning attacks corrupt the training data, leading AI models to learn incorrect behaviors [8]. Examples of this include label flipping [4] and backdoor attacks [5]. Evasion attacks, on the other hand, deceive the AI model at inference time by presenting specially crafted input samples that cause misclassifications [9]. For instance, traditional adversarial techniques such as the projected gradient descent method (PGDM) [7], the square attack [10], and the method developed by Carlini & Wagner (C&W) [6], exemplify approaches that apply minor, often imperceptible, changes to input data, effectively misleading machine learning models into making erroneous classifications or decisions. Consequently, developing robust defenses to safeguard these AI models is critical to ensure their reliability and security. Adversarial training has been proposed as a method to improve robustness against specific attacks. However, while effective to some extent, supervised learning techniques, including adversarial training, may still be vulnerable to unknown test-time evasion (TTE) attacks that were not included in the training set [11]. Semi-supervised or unsupervised anomaly detection provides a practical alternative as it does not require labeling all the data. These methods can be utilised in various scenarios, improving the model's adaptability and generalisation to new and unseen data types, thus lowering the costs associated with data preparation. Directly benefiting from conventional anomaly detection methods to address adversarial attacks on images however remains challenging for three primary reasons. First, adversarial attacks are intentionally designed to be imperceptible to the human eye, making the noise they introduce very subtle. In some advanced attacks, even a single pixel may be altered [12], resulting in only a slight difference between benign and compromised images [13]. Consequently, traditional methods

This work was supported by the Turing's Defence and Security programme through a partnership with the UK government in accordance to the framework agreement between GCHQ & The Alan Turing Institute.

struggle to distinguish these two distributions effectively [14]. Second, there is a lack of faithful, low-dimensional features for images, which affects the performance of conventional methods such as Gaussian Mixture Models (GMM) or Support Vector Machines (SVMs) with high-dimensional data [15]. This high dimensionality further complicates the application of traditional anomaly detection techniques. Third, adversarial attacks do not necessarily follow a particular form of distribution, such as Gaussian, as exemplified by the canonical assumptions of GMMs.

In this paper, we seek to resolve these challenges through embedding 2D-signature features of learned representations (of images) into a semi-supervised anomaly detection framework. We call our algorithm *2DSig-Detect* (Algorithm 4). Fundamentally, we define an anomaly score associated to an image based on its "distance" in embedding space to a dataset of instances labelled as normal or benign. We investigate two choices of distance, namely the conformance score and the covariance norm. We apply a threshold to this anomaly score to ascertain whether a given test instance is anomalous relative to the benign corpus. Our approach does not make any assumptions on the distribution of the data or adversarial attack, and through our usage of 2D-signatures and learned representations we obtain useful low-dimensional features. Moreover our algorithm has a free choice of distance metric. Combined, these mitigate some of the issues with conventional algorithms.

Below we outline our novel contributions in this paper:

- We introduce 2D-signature method which allows for the computation of both cross-channels and self-channels. Prior 2D-signature calculations only compute cross-channel terms and thus result in not being able to learn "gray-scale" information, i.e. information from correlations from the same R, G or B channel with itself.
- Compared with non-signature based methods, i.e. CNNs, our method allows for a compact description which separates out the cross-channel and self-channel information. CNNs automatically combine this information over the convolution kernel.
- As a proof of concept, we demonstrate that the 2D-signature method can be applied to mitigate against evasion and backdoor attacks. Our results are competitive with the spectral signature method [16] for mitigating backdoor attacks, whilst being significantly more computationally efficient.

1.1. 2D-signature from learned representations

The signature transform, based in rough path theory [17], can be used in machine learning as a feature transform on the underlying path of multi-modal streamed data [18]. The resulting signature is a graded infinite dimensional sequence composed of the coordinate iterated integrals of the path. The signature is a natural choice for the feature set of streamed data for three reasons: it captures cross-channel interactions, is an efficient summary of the data, and ignoring a pathological set, uniquely characterises the underlying path of the data.

With its efficiency in capturing the total ordering of data and summarising the data over segments, signature-based machine learning models have been proved efficient in several fields of application, from automated recognition of Chinese handwriting [19] to early-warning prediction of sepsis [20, 21] and the diagnosis of mental health problems [22, 23].

Recently, the signature transform has been generalised to higher orders [24, 25, 26], in particular to two dimensions. The 2D-signature transform fulfills a similar role for images as the signature does for time series. It provides a graded summary of image data (which may have an arbitrary number of channels) over their "area" information. The signature is graded in the sense that its sequence is delineated in levels, where each level captures increasing statistical variation of the path and surface for the 1D and 2D case respectively. One major advantage of using 2D-signature features is that the length of the 2D-signature at each level only depends on the number of channels, thus does not grow with resolution. This invariance means that the computational complexity of the 2D-signature transform is manageable, leading to consistent performance across varying resolutions. Moreover, 2D-signature features are more memory and compute efficient compared with the storage and processing costs of the original image. This is particularly important for applications with limited resources, such as mobile devices.

By expanding integrated integrals from 1D line elements to 2D plane elements, a freedom of choice of parameterisation across 2D elements allows for the computation of both cross-channel and self-channel terms. We note that in [24] a restriction to only computing 2D-signatures with cross channel terms results in not being able to learn gray-scale information. We compute both sets of terms in our implementation. Our 2D-signatures therefore have the advantage of compactly being able to separate out the contribution

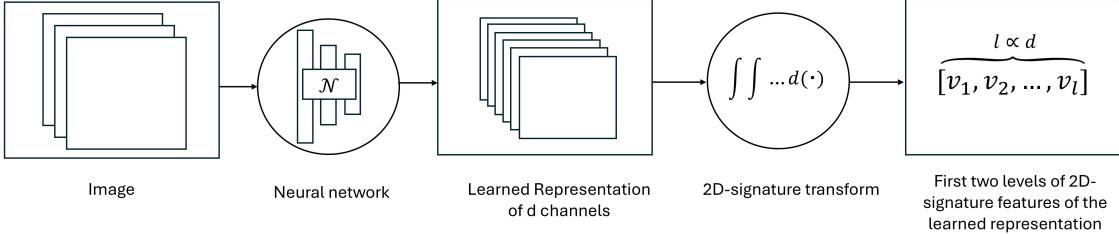


Figure 1: Process of Learned Representations Extraction and 2D-signature Calculation.

from information coming from the same channel to information coming from different channels, where conventional CNN models automatically combine this information.

Learned representations refer to high-dimensional projections of data that encode relevant features or patterns inferred by a model from unlabelled data. These representations have proven efficient in helping to separate distributions when they are mixed [16]. This motivates using learned representations and 2D-signatures together: *first we lift the data to a higher-dimensional feature space of representations where the data is better separated, and then we use 2D-signatures to extract faithful and compact features.* The simplified process flow is shown in Figure 1.

Our algorithm is inspired by [27] and [28]; the former presents an anomaly detection method for sequential data, in which the authors introduce the Mahalanobis-distance-based *conformance score* defined on the range of the signature map. The conformance score gives the minimum distance between a sample and a set (in our case, this is in this space of signature features). Subsequently, [28] applied this method to detect radio-frequency interference in radio-astronomy signals. We propose a modified anomaly detection algorithm, called 2DSig-Detect, that adapts and extends their approach. We choose to use a Mahalanobis-equivalent distance called the covariance norm. We do not follow [27] and use the conformance score because the computational cost of their method increases with the size of the training set, raising issues when applied to tasks like online anomaly detection where swift sensor responses and low latency are crucial priorities. To demonstrate the advantage of our choice of distance we compare the performance of 2DSig-Detect under the conformance score and covariance norm, respectively.

We establish conditions under which our algorithm achieves both low false negative and false positive rates, providing theoretical evidence for the effectiveness of 2DSig-Detect. The effectiveness of 2DSig-Detect is evaluated against the same framework but with the conformance score (instead of the covariance norm). In addition, we evaluate our approach by benchmarking against a GMM implementation and the naive-norm. Our results consistently demonstrate superior performance of 2DSig-Detect in terms of both time cost and multiple metrics, such as false negative rate, F1-score, and area under the receiver operating characteristic curve (AUC-ROC).

1.2. Paper organisation

This paper is organized as follows: Section 2 presents our generalised framework for anomaly detection. In particular we introduce the covariance norm and conformance score in 2.1, and following this in 2.2 detail how to integrate them into a semi-supervised framework given in Algorithm 3 alongside theoretical error bounds for our chosen distance metrics in Proposition 2.4 and Proposition 2.5. Section 2.3 and 2.4 describe the 2D-signature and learned representations, and then Section 2.5 explains how these are integrated together in our proposed algorithm, 2DSig-Detect, shown in Algorithm 4.

After introducing our algorithm, we present experimental results in Section 3. We describe the datasets used in the experiments, including CUReT and CIFAR-10 (Section 3.1), and perform three experiments:

- Texture classification (Section 3.2)
- Testset-level defence i.e. evasion attacks (Section 3.3)
- Trainset-level defence i.e. backdoor attacks (Section 3.4)

For each experiment we explain the setup used to evaluate the performance of 2DSig-Detect, including against benchmarks. We highlight the the effectiveness of our model at detecting these different types of attacks. In addition to this, in Section 3.3.4 we explain the role of learned representations, using the context of evasion attacks to exemplify this.

Finally, Section 4 concludes the paper, summarising our key findings and suggesting directions for future work in this area.

2. Methods

In this section, we provide a comprehensive description of the underlying metrics that we will use to develop our 2D-signature based anomaly detection scores, along with a clear definition of how outliers are detected and an explanation of why we view one framework as preferable over the other.

2.1. Underlying metrics for anomaly scores

We consider two choices of metric for 2DSig-Detect, namely the covariance norm and the conformance score. These result in two variants of 2DSig-Detect which we refer to as *2DSig-Norm* (for the covariance norm) and *2DSig-Conf* (for the conformance score). The choice of metric effects the computational cost of the algorithm, which influences our choice between them. This subsection introduces the covariance norm and conformance score metrics, explains why they are suitable for anomaly detection tasks, and describes how they can be computed. We defer a comparison between their computational costs to the next subsection, where we introduce the general framework in which we use these metrics to find anomalies.

Algorithm 1 function *covariance_norm*.

- 1: **input:** $x \in \mathbb{R}^d$, a single input data;
 - 2: \mathcal{C} , a finite corpus of data with $|\mathcal{C}|$ number of elements, and its distribution is $\mathcal{L}_{\mathcal{C}}$.
 - 3: **initialization:** a $d \times d$ covariance matrix $A = 0$.
 - 4: **output:** $\|x\|_{\mathcal{L}_{\mathcal{C}}}$, the covariance norm of x under $\mathcal{L}_{\mathcal{C}}$.
 - 5: Set $\mu = \frac{1}{|\mathcal{C}|} \sum_{y \in \mathcal{C}} y$.
 - 6: Let $\bar{Y} = [y - \mu]_{y \in \mathcal{C}}$, an $|\mathcal{C}| \times d$ matrix.
 - 7: Set $A = \frac{1}{|\mathcal{C}| - 1} \bar{Y}^T \bar{Y}$.
 - 8: **return:** $\|x\|_{\mathcal{L}_{\mathcal{C}}} := \sqrt{(x - \mu)^T A^{-1} (x - \mu)}$.
-

Definition 2.1 (Covariance norm). *Assume that there is a centred distribution \mathcal{L} on a vector space V with a finite second moment. Denote by V' the dual of V . The covariance quadratic form $\text{cov}(f, g) := \mathbb{E}^{\mathcal{L}}[f(\cdot)g(\cdot)]$, where $f, g \in V'$, induces a dual norm defined for $x \in V$ by*

$$\|x\|_{\mathcal{L}} = \sup_{f: \text{cov}(f, f) \leq 1} f(x). \quad (1)$$

As suggested by [27], this norm in theory splits V into two subspaces, based on whether x is on the linear span of the support of \mathcal{L} , denoted by $\text{span}(\text{supp}(\mathcal{L}))$: $\|x\|_{\mathcal{L}} < \infty$ when $x \in \text{span}(\text{supp}(\mathcal{L}))$, and $\|x\|_{\mathcal{L}} = \infty$ when $x \notin \text{span}(\text{supp}(\mathcal{L}))$. This can be seen as a natural tool for detecting outliers for distributions, as outliers typically deviate significantly from the origin (or the mean), they tend to have large, if not infinite, norm values. Note that the covariance norm coincides with the *Mahalanobis distance* [29] when V is the Euclidean space and \mathcal{L} with invertible covariance matrix. When $V = \mathbb{R}^d$, $d \in \mathbb{N}$, we are able to derive an explicit formula for Definition 2.1.

Proposition 2.2. *Suppose that there is a centred distribution \mathcal{L} on \mathbb{R}^d with a finite second moment, then for $x \in \mathbb{R}^d / \{0\}$*

$$\|x\|_{\mathcal{L}}^2 = \langle x, A^{-1}x \rangle,$$

where $A_{i,j} = \text{cov}(e_i, e_j)$ and $(e_i)_{1 \leq i \leq d}$ is a canonical basis for \mathbb{R}^d .

In practice, given a corpus of data, say \mathcal{C} , the covariance norm of an incoming instance x to \mathcal{C} can be computed through Algorithm 1. For data $X \in \mathbb{R}^{n \times d}$, a matrix with n input data vectors, the $\{\|x\|_{\mathcal{L}_{\mathcal{C}}}\}_{x \in X} := \sqrt{\text{diag}((X - \mu)A^{-1}(X - \mu)^T)}$, where $\text{diag}(M)$ represents the diagonal elements of M , and (μ, A) is pair of the empirical mean and covariance of \mathcal{C} .

Previous literature [27] further introduce the following notation for anomaly detection purpose.

Definition 2.3 (Conformance score). *Suppose that there is a distribution \mathcal{L} on a vector space V with a finite second moment. The the conformance score of x to \mathcal{L} is defined as the distance*

$$\text{dist}(x; \mathcal{L}) := \inf_{y \in \text{supp}(\mathcal{L})} \|y - x\|_{\mathcal{L}}.$$

Algorithm 2 function *conformance_score*.

- 1: **input:** $x \in \mathbb{R}^d$, a single input data;
 - 2: \mathcal{C} , a finite corpus of data with $|\mathcal{C}|$ number of elements, and its distribution is $\mathcal{L}_{\mathcal{C}}$.
 - 3: **output:** $\text{dist}(x; \mathcal{L}_{\mathcal{C}})$, the conformance score of x to corpus \mathcal{C} .
 - 4: **initialization:** a $d \times d$ covariance matrix $A = 0$.
 - 5: set $\mu = \frac{1}{|\mathcal{C}|} \sum_{y \in \mathcal{C}} y$;
 - 6: Let $\bar{Y} = [y - \mu]_{y \in \mathcal{C}}$ and $Y = [y - x]_{y \in \mathcal{C}}$ two $|\mathcal{C}| \times d$ matrices.
 - 7: Set $A = \frac{1}{|\mathcal{C}| - 1} \bar{Y}^T \bar{Y}$.
 - 8: Set $\text{dist}(x; \mathcal{L}_{\mathcal{C}}) = \sqrt{\min(\text{diag}(YA^{-1}Y^T))}$.
 - 9: **return:** $\text{dist}(x; \mathcal{L}_{\mathcal{C}})$.
-

The conformance score traces the closest point to x in the closed support of \mathcal{L} and computes the corresponding variance distance (norm). We observe that $\text{dist}(x; \mathcal{L}) = 0$ if $x \in \text{supp}(\mathcal{L})$ and $\text{dist}(x; \mathcal{L}) = \infty$ if $x \notin \text{span}(\text{supp}(\mathcal{L}))$. Indeed, the conformance score is mainly used to quantify the distance of x to the distribution \mathcal{L} when $x \in \text{span}(\text{supp}(\mathcal{L})) - \text{supp}(\mathcal{L})$. An example is an empirical distribution associated to a finite set of observations $\{y_i : y_i \in V, 1 \leq i \leq N\}$. When $N \gg \dim(V)$, then it is possible that the linear span of the support of this empirical distribution is V itself. In practice, given a corpus of data, say \mathcal{C} , the covariance norm of an incoming instance x to \mathcal{C} can be computed through Algorithm 2.

2.2. Frameworks for anomaly scores

We may adopt either Definition 2.1 or Definition 2.3 to measure the “distance” of each input x to the underlying distribution of \mathcal{C} : the shorter the distance is, the less likely this instance is an outlier. Reasonable thresholds are needed for making a decision after obtaining the covariance norm of x (resp. the conformance score of x to $\mathcal{L}_{\mathcal{C}}$): whether the score is high enough so that x can be classified as anomaly. We will refer to them as the *cov-norm framework* and the *conf-score framework* throughout this paper. Algorithm 3 describes one way to determine the threshold within \mathcal{C} , where a list of scores is generated and sorted on the validation set, and the threshold is set to be the r th percentile of the list, denoted by α_r .

Algorithm 3 function *threshold*.

- 1: **input:** \mathcal{C} , a finite corpus of data with distribution $\mathcal{L}_{\mathcal{C}}$;
 - 2: r , r th percentile.
 - 3: **output:** thresholds α_r which is r th percentile;
 - 4: \mathcal{C}_t , the reference corpus.
 - 5: randomly split \mathcal{C} evenly into two sets, \mathcal{C}_t and \mathcal{C}_v .
 - 6: **initialization:** an empty list l .
 - 7: **for** each $x \in \mathcal{C}_v$ **do**
 - 8: append $\text{covariance_norm}(x, \mathcal{C}_t)$ of Algorithm 1 (resp. $\text{conformance_score}(x, \mathcal{C}_t)$ of Algorithm 2) to l ;
 - 9: **end for**
 - 10: **sort** l in an ascending order.
 - 11: Find the value in the list l that is r th percentile.
 - 12: **return:** α_r and \mathcal{C}_t .
-

Algorithm 3 also provides a reference corpus for measuring distance of unseen data point. Note that Algorithm 3 can be applied with various distances by replacing Line 8 with the custom distance. Regarding the computational cost, by construction, the conf-score framework is n th-fold of that of the cov-norm framework if n is the number of samples in the training set.

In the following, we present some useful probability bounds of Type-I and Type-II errors (false positive and false negative) for both frameworks. The proofs are postponed to Appendix A.

Proposition 2.4 (Bounds for Type-I and Type-II errors of cov-norm framework). *Suppose \mathcal{L}_c is a distribution with mean and covariance pair (μ, A) . Assume that A is finite and invertible. Consider the covariance norm $\|x - \mu\|_{\mathcal{L}_c}$, where x may be drawn from \mathcal{L}_c , or drawn from an unknown distribution \mathcal{L}_u , with mean and covariance pair (μ_u, A_u) . Assume that $\sqrt{d} \ll |A^{-\frac{1}{2}}(\mu - \mu_u)|/\|A^{-1/2}A_u^{1/2}\|$, where $\|B\|$ represents the matrix operator norm of a matrix B .*

Then for a predetermined $\delta \in (\sqrt{d}, |A^{-\frac{1}{2}}(\mu - \mu_u)|)$ such that

$$\sqrt{d} \ll \min \left(\delta, \frac{|A^{-\frac{1}{2}}(\mu - \mu_u)| - \delta}{\|A^{-1/2}A_u^{1/2}\|} \right),$$

it holds that

$$\mathbb{P}(\|x - \mu\|_{\mathcal{L}_c} > \delta \mid x \sim \mathcal{L}_c) \leq \frac{d}{\delta^2} \quad (2)$$

and

$$\mathbb{P}(\|x - \mu\|_{\mathcal{L}_c} < \delta \mid x \sim \mathcal{L}_u) \leq \frac{d\|A^{-\frac{1}{2}}A_u^{\frac{1}{2}}\|^2}{(|A^{-\frac{1}{2}}(\mu_u - \mu)| - \delta)^2}. \quad (3)$$

In particular, when $\delta = \frac{|A^{-\frac{1}{2}}(\mu_u - \mu)|}{1 + \|A^{-\frac{1}{2}}A_u^{\frac{1}{2}}\|}$,

$$\mathbb{P}(\|x - \mu\|_{\mathcal{L}_c} < \delta \mid x \sim \mathcal{L}_u) \leq \frac{d}{\delta^2}. \quad (4)$$

Proposition 2.5 (Bounds for Type-I and Type-II errors of conf-score framework). *Given a corpus \mathcal{C} (\mathbb{R}^d -valued) consisting of n samples drawn independently from a distribution \mathcal{L}_c , with ground truth mean and covariance (μ, A) . Assume that A is invertible. Given the unknown instance x and consider the conformance score $\text{dist}(x; \mathcal{L}_c)$.*

For a pair (d, δ) such that

$$\sqrt{2d} < \delta < |A^{-\frac{1}{2}}(\mu_u - \mu)| \text{ and } \frac{d\|\Delta_A^{\frac{1}{2}}\|^2}{(\Delta_\mu - \delta)^2} < 1,$$

where (μ_u, A_u) are the mean and covariance pair for the unknown distribution \mathcal{L}_u , and $\Delta_A := A^{-\frac{1}{2}}A_uA^{-\frac{1}{2}} + I$, then

$$\mathbb{P}(\text{dist}(x; \mathcal{L}_c) > \delta \mid x \sim \mathcal{L}_c) \leq \left(\frac{2d}{\delta^2} \right)^n, \quad (5)$$

and

$$\mathbb{P}(\text{dist}(x; \mathcal{L}_c) < \delta \mid x \sim \mathcal{L}_u) \leq 1 - \left(1 - \frac{d\|\Delta_A^{\frac{1}{2}}\|^2}{(\Delta_\mu - \delta)^2} \right)^n. \quad (6)$$

Proposition 2.5 gives an increasing up-bound for Type-II error, which is not expected. Note that

$$\|\mu_u\|_{\mathcal{L}_c} = |A^{-1/2}(\mu_u - \mu)|, \quad (7)$$

the *mean difference* under the covariance norm of \mathcal{L}_c , appears in both propositions. Note that under certain conditions, the mean difference appears in the formula of Kullback-Leibler divergence [30]. Proposition 2.4 and Proposition 2.5 both suggest that the key features for these two frameworks to work well are the dimensionality as well as how separated the two distributions are (mainly measured by mean difference).

Note that we prefer the covariance norm over the conformance score because under some circumstances, it will give similar performance but the former saves computational costs. In the following, we will introduce a transform on images to capture their distribution while reducing dimensionality (Section 2.3), and find ways to leverage the distribution difference (Section 3.3.4).

2.3. 2D-Signature

We will give a brief introduction to 2D-signature features for color images and generalise them for high-dimensional representations. Define $[n] := \{1, 2, \dots, n\}$. For an image of size (N, M, d) , with $N, M, d \in \mathbb{N}$, i.e. with $M \times N$ many pixels and d many channels, the image data is defined as follows.

Definition 2.6 (Continuous and discrete image data). *For $N, M, d \in \mathbb{N}$, the image information can be represented via a map $\mathbf{x} : \{1, \dots, N\} \times \{1, \dots, M\} \rightarrow [0, 1]^d$, where d represents the number of channels of an image, \mathbf{x}_{k_1, k_2}^i denotes the value of i -th coordinate of the image, $i \in [d]$, at the location of pixels (k_1, k_2) with $k_1 \in [N]$ and $k_2 \in [M]$. Through interpolation, the (normalised) image data can be well-represented via a piecewise continuous map $x : [0, 1]^2 \rightarrow [0, 1]^d$, where $x^i(s, t)$ represents the value of i -th coordinate of the image at the location (s, t) . We denote the space of image data as \mathcal{Z}_d .*

Note that (N, M, \cdot) is also known as image resolution. Definition 2.6 coincides with color images when $d = 3$, where the three coordinates represents red/green/blue (RGB) channels respectively, and reduces to gray images when $d = 1$.

Image data is parameterised by two components, one for each dimension of the data, which are referred as s and t in the following context.

The differentials and the definition The key components of 2D-signature are the area increments we want to examine for image data. We shall adopt the notation of [31] for compact expressions:

$$dx_{s,t}^i := \frac{\partial^2 x^i(s,t)}{\partial s \partial t} ds dt, \quad i \in [d], \quad (8)$$

$$\hat{dx}_{s,t}^{ij} := \frac{\partial x^i(s,t)}{\partial s} \frac{\partial x^j(s,t)}{\partial t} ds dt, \quad i, j \in [d]. \quad (9)$$

Both differentials provide different insights into the behavior of the function $x^i(s,t)$ in the s - t plane: the first differential $dx_{s,t}^i$ measures how the rate of change with respect to one variable varies as the other variable changes and is sensitive to the curvature or bending of the function surface in the s - t plane; and the second one measures the interaction of the independent rates of change in s and t , and captures how changes in one direction affect changes in the other direction multiplicatively. The choice of these two differentials comes from the fact that, for any sufficiently smooth function $f : \mathbb{R}^d \rightarrow \mathbb{R}$, if considering the rectangle increment of it over $[s_1, s_2] \times [t_1, t_2]$, then

$$\begin{aligned} & f(x_{s_2, t_2}) - f(x_{s_1, t_2}) - f(x_{s_2, t_1}) + f(x_{s_1, t_1}) \\ &= \sum_i \int \int_{[s_1, s_2] \times [t_1, t_2]} \frac{\partial f(x_{s,t})}{\partial x^i} dx_{s,t}^i + \sum_{i,j} \int \int_{[s_1, s_2] \times [t_1, t_2]} \frac{\partial^2 f(x_{s,t})}{\partial x^i \partial x^j} \hat{dx}_{s,t}^{ij}. \end{aligned} \quad (10)$$

Equivalently, the area feature of f acting on image x can be represented through the linear combination of $dx_{s,t}^i$ and $\hat{dx}_{s,t}^{ij}$. As the integrals with respect to $dx_{s,t}^i$ or $\hat{dx}_{s,t}^{ij}$ can be computed, it would be expected that through a Taylor expansion, the area effect of an arbitrary f can be approximated as

$$\begin{aligned} & f(x_{s_2, t_2}) - f(x_{s_1, t_2}) - f(x_{s_2, t_1}) + f(x_{s_1, t_1}) \\ & \approx \sum_i c_i \int \int_{[s_1, s_2] \times [t_1, t_2]} dx_{s,t}^i + \sum_{i,j} C_{i,j} \int \int_{[s_1, s_2] \times [t_1, t_2]} \hat{dx}_{s,t}^{ij} \\ & \quad + \sum_{i,j} c_{i,j} \int_{s_1 < v < s < s_2} \int_{t_1 < u < t < t_2} dx_{v,u}^j dx_{s,t}^i \\ & \quad + \sum_{i,j,\bar{i},\bar{j}} C_{i,j,\bar{i},\bar{j}} \int_{s_1 < v < s < s_2} \int_{t_1 < u < t < t_2} \hat{dx}_{v,u}^{ij} \hat{dx}_{s,t}^{ij} \\ & \quad + \dots, \end{aligned} \quad (11)$$

where the constants are to be determined through learning from the data. However, as commented in [31], the number of terms will explode when we expand the Taylor series, this explains why we are not aware of any extension up to an arbitrary order in literature. We therefore would follow [31], restricting to the differential

$$\mathbf{dx}_{s,t} := \begin{bmatrix} dx_{s,t}^1 \\ \dots \\ dx_{s,t}^d \\ \hat{dx}_{s,t}^{11} \\ \hat{dx}_{s,t}^{22} \\ \dots \\ \hat{dx}_{s,t}^{dd} \end{bmatrix}, \quad (12)$$

and define the 2D-signature up to the second level only.

Definition 2.7 (2D-signature for images, up to the second level). *Let $x : [0, 1]^2 \rightarrow [0, 1]^d$ be continuous image data such that the following integration makes sense. For any closed intervals $J_1, J_2 \subset [0, 1]$, define*

$$\begin{aligned} & (S_{J_1 \times J_2}(x))^p \\ &= \int \dots \int_{\substack{s_1 < \dots < s_n, s_i \in J_1 \\ t_1 < \dots < t_n, t_i \in J_2}} \mathbf{dx}_{s_1, t_1} \otimes \dots \otimes \mathbf{dx}_{s_n, t_n} \in (\mathbb{R}^{2d})^{\otimes p}, \end{aligned} \quad (13)$$

and set

$$S_{J_1 \times J_2}(x) = (1, S_{J_1 \times J_2}^1(x), S_{J_1 \times J_2}^2(x)) \quad (14)$$

while $S_{J_1 \times J_2}^p(x) = (S_{J_1 \times J_2}(x))^p$ is the degree p or level p of the signature. We write $S(x)$ short for $S_{J_1 \times J_2}(x)$ when the intervals are obvious or when we do not need to specify.

For instance, when $J_1 = J_2 = [0, 1]$ and $p = 1$,

$$S^1(x) = \int \int_{[0,1]^2} \mathbf{d}x_{s,t} = \begin{bmatrix} \int \int_{[0,1]^2} dx_{s,t}^1 \\ \vdots \\ \int \int_{[0,1]^2} dx_{s,t}^d \\ \int \int_{[0,1]^2} \hat{dx}_{s,t}^{11} \\ \vdots \\ \int \int_{[0,1]^2} \hat{dx}_{s,t}^{dd} \end{bmatrix}.$$

The definition coincides with 2D-id-signature in [26] if $\mathbf{d}x_{s,t}$ in Eqn. (12) only include the first half, i.e., $\{\mathbf{d}x_{s,t}^i\}_{i=1}^d$, where the extension can be made up to an arbitrary order and its algebraic structures are further explored. We do not follow [26] because the integrals involving $\hat{dx}_{s,t}^{ii}$ capture nonlinear effect of pixels from the first level (see **Discretization** below) and play a vital role in learning tasks [31].

The number of features in Eqn. (14) would be $2d$ for level 1, and $(2d)^2$ for level 2. For image-related tasks, especially in later layers, where image dimensions have often been reshaped to have more than their 3 initial layers, interaction between channels is not so crucial for our learning purpose [32]. We may consider the components that reinforces the effect on the same channel in the second level, i.e., we consider $dx^i dx^i$ and $\hat{dx}^{ii} \hat{dx}^{ii}$. Thus the number of features would be $2d$ as well for this special collection of level 2.

Discretization As we have two basic notations, we shall give examples on level 1 only for illustration. Note that for any $0 \leq s_1 < s_2 \leq 1$ and $0 \leq t_1 < t_2 \leq 1$, it holds that

$$\int_{[t_1, t_2]} \int_{[s_1, s_2]} \frac{\partial^2 x^i(s, t)}{\partial s \partial t} ds dt = x^i(s_2, t_2) - x^i(s_2, t_1) - x^i(s_1, t_2) + x^i(s_1, t_1). \quad (15)$$

Therefore

$$\int \int_{[0,1]^2} \frac{\partial^2 x^i(s, t)}{\partial s \partial t} ds dt = \mathbf{x}_{N,M}^i - \mathbf{x}_{N,1}^i - \mathbf{x}_{1,M}^i + \mathbf{x}_{1,1}^i. \quad (16)$$

For a small stepsize $h \ll 1$, using forward difference gives

$$\begin{aligned} & \frac{\partial x^i(s, t)}{\partial s} \frac{\partial x^i(s, t)}{\partial t} ds dt \\ & \approx (x^i(s+h, t) - x^i(s, t))(x^i(s, t+h) - x^i(s, t)). \end{aligned} \quad (17)$$

Thus

$$\begin{aligned} & \int \int_{[0,1]^2} \frac{\partial x^i(s, t)}{\partial s} \frac{\partial x^i(s, t)}{\partial t} ds dt \\ & \approx \sum_{k_1=1}^{N-1} \sum_{k_2=1}^{M-1} \left((\mathbf{x}_{k_1+1, k_2}^i - \mathbf{x}_{k_1, k_2}^i)(\mathbf{x}_{k_1, k_2+1}^i - \mathbf{x}_{k_1, k_2}^i) \right). \end{aligned} \quad (18)$$

Clearly, Level 1 of the 2D-signature contains both linear effects and nonlinear effects of pixel information. Level 2 further reinforces these effects through nonlinear operations (iterated integrals). One may find the discretization for level 2 of the 2D-signature in Appendix B.

Note that there exists another type of 2D-signature, called *Type II* 2D-signature in this paper, whose expression can be derived by iterating the Jacobian minor as defined in [24]. Type II 2D-signature is not considered in this article because its expression focuses solely on cross-channel interactions, thereby excluding its application to grayscale images. However, when a d -channel image is expanded to a $(d+2)$ -channel image by padding at positions (s, t) , it is claimed that the corresponding Type II 2D-signature feature has the characteristic property: the law of random images can be completely determined by the expected 2D-signature of the expanded images (see [24, Theorem 6]).

2.4. Learned representations

It has been shown that learned representations (e.g. via a neural network) may better separate distributions [16]. Inspired by this observation, in practice we may consider lifting the space where a corpus \mathcal{C} lives in to a higher-dimensional space $\mathcal{R}(\mathcal{C})$, where \mathcal{R} is a carefully-chosen learned representation and $\mathcal{R}(\mathcal{C}) := \{\mathcal{R}(y), y \in \mathcal{C}\}$.

Later in Section 3.3.4, we explore the capability of different learned representations in separating distributions. For a well-trained classification model, it's expected that the later layers are adept at distinguishing different image clusters. In contrast, for anomaly detection, the earlier layers of this trained model are more effective at emphasising the differences between benign and polluted images, rather than the later layers.

Algorithm 4 function *2DSig-Detect*.

- 1: **input:** \mathcal{C} , a finite corpus of data with distribution \mathcal{L}_C ;
- 2: \mathcal{T} , a test set of data;
- 3: \mathcal{N} , a neural network model providing feature representation \mathcal{R}_ϕ ;
- 4: \mathcal{S} , other transform such as 2D-signature;
- 5: r , rth percentile.
- 6: **output:** \mathcal{T}_b , a subset of \mathcal{T} .
- 7: extract features $(\mathcal{S} \circ \mathcal{N})(\mathcal{C})$ of \mathcal{C} .
- 8: apply Algorithm 3 to $(\mathcal{S} \circ \mathcal{N})(\mathcal{C})$: randomly split $(\mathcal{S} \circ \mathcal{N})(\mathcal{C})$ evenly into two sets, $(\mathcal{S} \circ \mathcal{N})(\mathcal{C})_t$ and $(\mathcal{S} \circ \mathcal{N})(\mathcal{C})_v$, and get a threshold α_r .
- 9: extract features $(\mathcal{S} \circ \mathcal{N})(\mathcal{T})$ of \mathcal{T} .
- 10: initialise an empty set \mathcal{T}_b .
- 11: **for** each $x \in \mathcal{T}$ **do**
- 12: assign a score s_x to x by applying either Algorithm 1 or Algorithm 2 to $(\mathcal{S} \circ \mathcal{N})(x)$ and $(\mathcal{S} \circ \mathcal{N})(\mathcal{C})_t$;
- 13: **if** $s_x < \alpha_r$ **then**
- 14: $\mathcal{T}_b \leftarrow \mathcal{T}_b + \{x\}$;
- 15: **end if**
- 16: **end for**
- 17: **return:** \mathcal{T}_b .

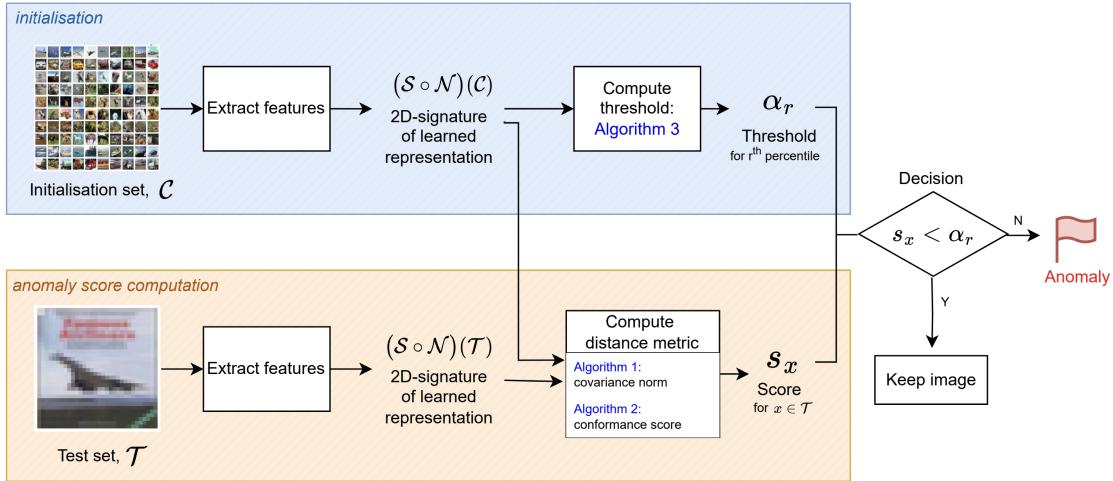


Figure 2: Flowchart of the proposed 2DSig-Detect framework, as outlined in Algorithm 4.

2.5. 2DSig-Detect

Considering its ability in capturing effects of image data, the proposed methods are therefore embedding 2D-signature features in the base frameworks: we extract a higher-dimensional representation of an image x through a neural network transform \mathcal{R} , then apply the 2D-signature transform S^i ($i = 1$ or 2) to $\mathcal{R}(x)$, and finally feed these 2D-signature features $S^i(\mathcal{R}(x))$ into one of the two base frameworks (Algorithm 1 or 2) to get a score. The threshold can be obtained via applying Algorithm 3 to the transformed train set $(S^i \circ \mathcal{R})(\mathcal{C})$. Together, this results in Algorithm 4, which we call *2DSig-Detect*. The flowchart of this process is summarised and presented in Fig. 2.

Throughout this paper, we refer to 2DSig-Detect (Algorithm 4) with the covariance norm and conformance score as *2DSig-Norm* and *2DSig-Conf*, respectively. We emphasize that *2DSig-Norm* and *2DSig-Conf* are versions of the same algorithm: 2DSig-Detect. The only change is the choice of distance metric that governs the resulting anomaly score. We propose, for computational efficiency reasons that

the experimental section (Section 3) will make clear, to favour *2DSig-Norm* over *2DSig-Conf*; instead we use the latter as a benchmark. In addition to this, we benchmark against our base framework (Algorithm 3) with flattened pixel features, which we refer to as the *naive-norm* and *naive-conf* approach, respectively. We also compare against the Gaussian mixture method with 2D-signature features, termed *2DSig-GMM*.

3. Applications

In this section, we show how the proposed 2DSig-Norm can be used in applications, including classification and defence against both backdoor and evasion attacks, and present the performance of 2DSig-Norm in comparison to the 2DSig-Conf and naive-norm¹ frameworks using standard image datasets. All experiments were conducted on a server running Ubuntu 22.04.4 LTS (Jammy Jellyfish) 64-bit. The server features 755.3 GiB of memory and an Intel® Xeon® Gold 6140 CPU @ 2.30GHz with 72 cores, providing a robust computational environment for the applications.

3.1. Datasets

CUReT (Columbia-Utrecht Reflectance and Texture Database [33]) is a data set of 5612 texture images in 61 classes. As the original images contain black boundaries, they are manually cropped to a size of 240×320 . Figure 3 shows examples of the first 30 images from the CUReT database. Following [31], 42 textures from this dataset are used for our first experiment.



Figure 3: The first 30 images from the CUReT database.

Cifar10 [34] is a dataset consisting of 60,000 $(32,32,3)$ color images in 10 classes, with 6,000 images per class. The dataset is divided into 50,000 training images and 10,000 test images. The 10 classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. CIFAR-10 is a standardized and widely used dataset in machine learning and computer vision research [35][36]. It provides a diverse set of images, making it a valuable resource for evaluating algorithms. Figure 4 shows randomly selected examples from each class in the CIFAR-10 dataset.



Figure 4: Randomly selected examples from each class in the CIFAR-10 dataset.

3.2. Texture image classification

The 2D-signature transform achieves a superior performance in texture image classification on CUReT as a informative feature [31]. In the light of this observation, we identify texture image classification as a special anomaly detection problem: if we choose the nth texture, i.e. class 42 as the target texture, the remaining 41 kinds of textures would be treated as anomalous to the target class.

Data processing To prevent overfitting, each of these 42 textures is divided into left and right halves,

¹The computational cost for naive-conf is sometimes very high so we ignore it in most of the experiments.

and one of the halves are randomly drawn, with one portion allocated for generating the training set, while the other half is reserved for testing purposes. We randomly sample 200 (resp. 100) images of size $(m \times m)$ from each train (resp. test) texture, $m = 32, 64$. Fig. 5 presents a selection of 64x64 patches extracted from Soleirolia plant class of the CUReT dataset. Among the 200 samples in each train class, half of them are randomly drawn as the validation set for generating the score distribution and determining the threshold α_r , for two different thresholds, $r = 80$ and $r = 90$ th percentiles. The blue curve in the bottom subplot of Fig. 7 gives an example of the cumulative distribution from 2DSig-Norm, where we can identify the 80th (resp. 90th) percentile as the threshold.

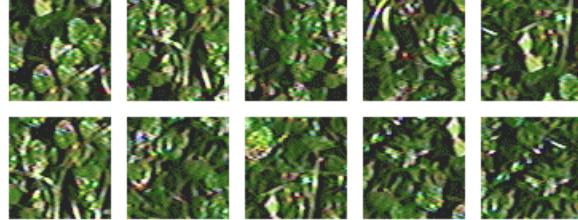


Figure 5: Examples of 64x64 patches from the Soleirolia plant class in the CUReT dataset.

3.2.1. Experiment set-ups

For each class, we generate three sets: scores from validation set (referred as *validation scores*), scores from test set of the same class (benign set), referred as *benign scores*, and scores from test sets of the remaining 41 classes (anomaly sets), referred as *anomaly scores*. The threshold shall be determined from the first set, and apply to the latter two. However, this introduces a highly unbalanced score set. For a fair comparison, we design two kinds of experiments for different comparison purposes.

Experiment 1. To compare performance between the two base frameworks introduced in Section 2.2, either with or without the 2D-signature, we will consolidate scores from all 42 validation sets into a unified validation score set. Similarly, we will get a unified benign score set and a unified anomaly score set. The threshold is determined on the unified validation score set, where one may sort the scores in an ascending order, and choose the r -th percentile as the threshold, and then apply it to the unified benign score set and the unified anomaly score set. Though the distributions of different classes on the training set may be different, Eqn. (2) (resp. Eqn. (6)) ensures that taking the covariance norm (resp. conformance score) is a way of *standardising* distributions: subtracting the dataset mean from a data point and then multiplied by the inverse of the covariance matrix.

Experiment 2. To compare performance between 2DSig-Norm and other unsupervised methods, we randomly selected 100 images from different classes in the segmented anomaly sets, so that the number of benign scores equals the number of anomaly scores. Anomaly detection is conducted per class and the mean metric plus standard deviation is recorded.

Metrics. In Experiment 1, F1-score (for $r = 80$ on validation), the area under the receiver operating characteristic curve (AUC) and time cost are used for model comparison. In Experiment 2, accuracy, F1-score and true positive rate (TPR) are used for model comparison. In this paper, “positives” refer to anomalies, while “negatives” refer to benign instances.

3.2.2. Results.

Experiment 1. For $m = 32$, we compare the proposed 2DSig-Norm against 2DSig-Conf as well as the naive-norm, in terms of F1-score (for $r = 80$), AUC and time cost. We examine the performance of level 1 and level 2 of the 2D-signature respectively. Each experiment recorded in Table 1 is repeated 10 times across different random seeds, where the mean and standard deviation are recorded. While both the 2DSig-Norm and the 2DSig-Conf exhibit significant advantages over naive-norm across all metrics, the 2DSig-Norm demonstrates a slight performance advantage over the 2DSig-Conf in both F1-score and AUC metrics, achieving this with only 1/7 of the time required by 2DSig-Conf.

To understand the performance difference, the mean difference between classes is also examined for different features, namely, the flatten feature, i.e., the flattened pixel information, level 1 of 2D-signature and level 2 of the 2D-signature in Fig. 6. The heat map of level 1 of the 2D-signature shows the largest

Table 1: Performance comparison for different features via either conf-score or cov-norm framework on CUReT dataset with sampling image size (32, 32, 3). Note that F1-score is captured at α_r with $r = 80$ th percentile.

		F1-score	AUC	time (s)
2DSig-Norm	Level1	0.92 ± 0.00	0.88 ± 0.01	0.62 ± 0.00
	Level2	0.94 ± 0.01	0.86 ± 0.01	0.73 ± 0.01
2DSig-Conf	Level1	0.92 ± 0.00	0.87 ± 0.01	4.30 ± 0.00
	Level2	0.93 ± 0.01	0.85 ± 0.01	9.71 ± 0.01
naive-norm		0.76 ± 0.01	0.72 ± 0.01	716.20 ± 51.80

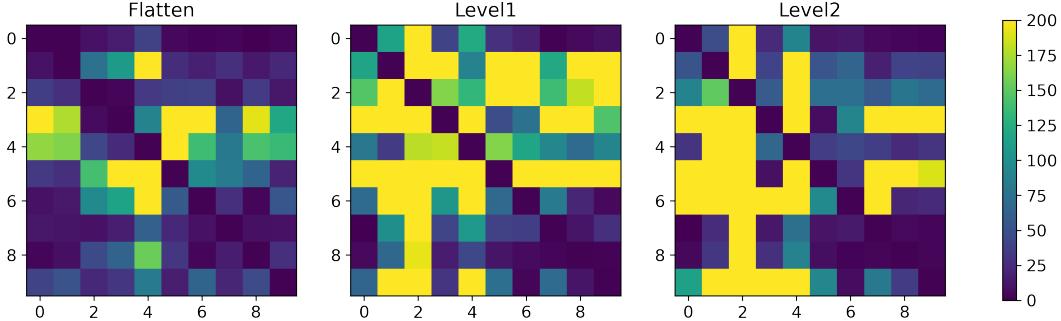


Figure 6: Mean differences calculated via (7) between the i th class of the training set and the j th class of the test set, $1 \leq i, j \leq 10$, on different features (Left: the flatten representation; Middle: the level 1 feature of 2D-signature; Right: the level 2 feature of 2D-signature).

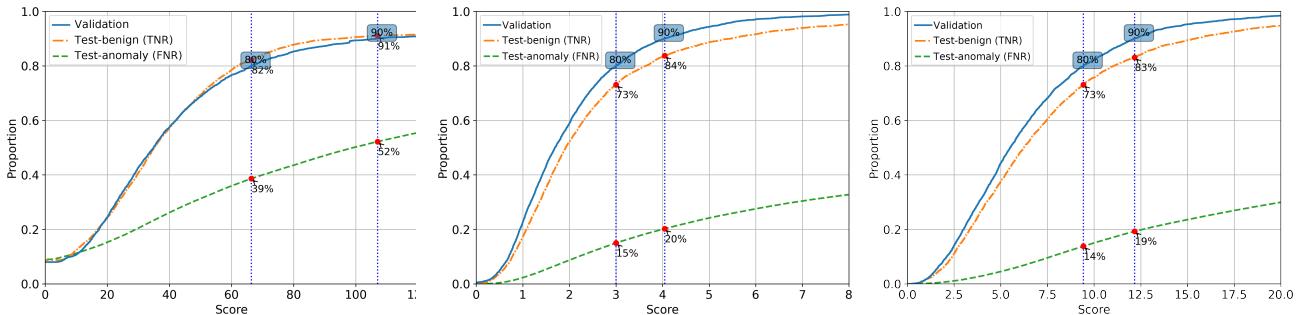


Figure 7: The impact on TNR and FNR of the test set when choosing different threshold (α_r with $r = 80$ th percentile and 90th percentile) of scores on validation set (TNR: percentage shown on the dashed-dotted orange curve; FNR: percentage shown on the dashed green curve) of three methods. From left to right are: naive-norm, 2DSig-Conf (Level1), and 2DSig-Norm (Level1).

discrepancy among classes, which coincides with Table 1, where under the same base framework, the AUC of level 1 is slightly higher than the one of level 2.

The ROC curves for Experiment 1 is shown in Fig. 7, demonstrating how the true negative rate (TNR) and false negative rate (FNR) change with different thresholds α_r , $r = 80$ th or 90th percentile. The naive-norm gives the worst FNR while achieving the highest TNR. Note that for the curves generated using the naive norm, there is a significant percentage of outliers (approximately ten percent) with scores of zero, leading to a non-zero lower bound for TNR. This indicates poor performance of the naive norm: regardless of how low the threshold is set, at least ten percent of outliers remain undetected. Compared to 2DSig-Conf, 2DSig-Norm achieves both higher TNR and lower FNR consistently. Fig. 8 gives corresponding ROC curves of Table 1. Note that the sharp jump of the ROC curve of naive-norm is due to the non-zero lower bound for TNR (and therefore FPR).

Compared to the the flatten representation, the models with 2D-signature features give better performance while significantly reducing the training time without loss of performance. We will focus more on comparing between 2DSig-Norm and 2DSig-Conf in the remaining part.

For $m = 64$, the results are recorded in Table 2.

Experiment 2. We compared two frameworks with Gaussian mixture model (GMM) [37] on level 1 of 2D-signature. GMM is chosen as a strong baseline for anomaly detection due to its ability to model

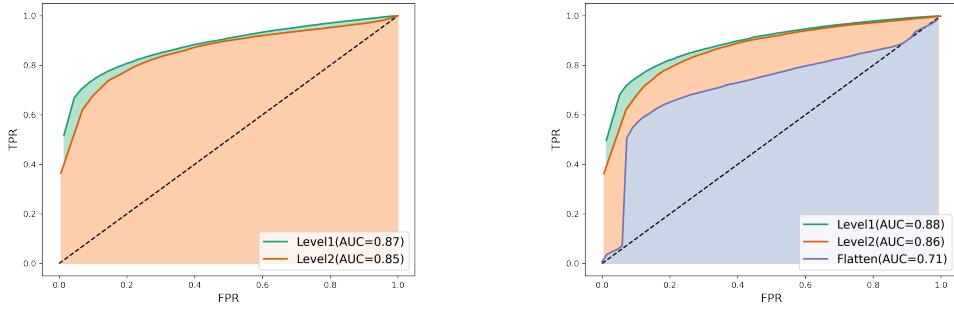


Figure 8: The ROC curves of three methods. Left: 2DSig-Conf on level 1 and level 2 2D-signature features; Right: 2DSig-Norm on level 1 and level 2 2D-signature features and naive-norm with the flatten representation feature.

Table 2: Comparison of the performance between 2DSig-Norm and 2DSig-Conf on the CUReT dataset with sampled image dimensions of (64, 64, 3). The metrics used for comparison are the mean \pm std of AUC and the time cost in seconds.

		AUC	time (s)
2DSig-Norm	Level1	0.91 \pm 0.01	0.65 \pm 0.07
	Level2	0.89 \pm 0.01	0.75 \pm 0.03
2DSig-Conf	Level1	0.91 \pm 0.02	4.11 \pm 0.07
	Level2	0.88 \pm 0.01	9.56 \pm 0.03

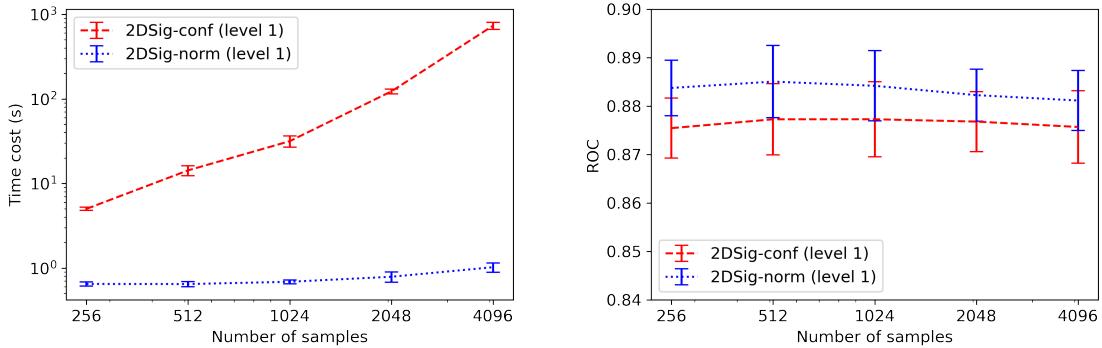


Figure 9: Time cost and AUC (mean \pm std) against number of samples in training set for 2DSig-Conf and 2DSig-Norm on the level 1 feature.

complex data distributions probabilistically and operate in an unsupervised manner, providing a flexible and adaptable approach to identifying anomalies [37]. The mean accuracy, TPR and F1-score (with standard deviations) are reported in Table 3. The 2DSig-Norm consistently outperforms the other two models on all three metrics. The TPR of 2DSig-Norm is twice that of the 2DSig-GMM.

Complexity To examine how the performance changes with the number of samples in the training set, we implement the first experiment via the 2DSig-Norm and the 2DSig-Conf (level 1 feature) on 2^j training samples, $j = 8, \dots, 12$ and plot the time cost as well as AUC (with standard deviation) against number of samples in Fig. 9. The time cost of the 2DSig-Conf and the 2DSig-Norm increase as polynomial and sub-linearly with respect to the number of samples respectively, while AUC stays at the same level for both. This again supports our choice: the 2DSig-Norm provides a cost-saving solution compared to the 2DSig-Conf.

Table 3: Performance comparison among 2DSig-GMM, 2DSig-Conf and 2DSig-Norm on CUReT dataset with sampled image dimensions of (32, 32, 3). The metrics used for comparison are the mean \pm std of accuracy, TPR and F1-score.

	Acc	TPR	F1-score
2DSig-GMM	0.675 \pm 0.100	0.458 \pm 0.238	0.637 \pm 0.126
2DSig-Conf	0.782 \pm 0.120	0.978 \pm 0.027	0.790 \pm 0.138
2DSig-Norm	0.794 \pm 0.128	0.983 \pm 0.027	0.803 \pm 0.135

3.3. Testset-level defense

The following experiments focus on detecting the presence of adversarial perturbations in images at test time. We aim to design a fast and accurate anomaly detector, using both 2DSig-Norm and 2DSig-Conf of the *2DSig-Detect*, to defend the trained model against both untargeted and targeted evasion attacks. The perturbed images are crafted using the iterative fast gradient sign method (IFGSM) [38] and the projected gradient descent method (PGDM) [39], two common evasion attack algorithms. The threat model considered assumes the adversary’s knowledge of the trained model and capability to manipulate data input at inference time, with the goal of producing an integrity violation [40]. We consider the image classification task on the CIFAR-10 dataset, using the pre-trained models RepVGG-A2 and ResNet-20, both achieving 92%+ classification accuracy. Table 4 shows that both IFGSM and PGDM give significantly high untargeted successful attack rate (SAR) and reasonable targeted SAR on ResVGG-A2².

3.3.1. Evasion attacks

Evasion attacks involve manipulating input data at test/inference time to deceive machine learning models through integrity violations. Adversarial examples are commonly used in these attacks, using methods such as IFGSM and PGSM, which are both rooted in gradient-based optimization techniques [41].

IFGSM is an attack method that perturbs input data by iteratively adding noise in the direction of the gradient of the loss function with respect to the input, scaled by a small step size. This small perturbation can significantly alter the model’s output, causing it to misclassify the input. Starting from the original image $\mathbf{x}_0 = \mathbf{x}$, the image is updated in the direction of the gradient of the loss function $\mathcal{L}(\theta, \mathbf{x}, l)$ with respect to the input image \mathbf{x} over multiple iterations N :

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h \cdot \text{sign}(\nabla_{\mathbf{x}_i} \mathcal{L}(\theta, \mathbf{x}_i, l)), i \in [N],$$

where h is a small step size, θ represents the model parameters, l is the true label of the image, and $\nabla_{\mathbf{x}_i} \mathcal{L}(\theta, \mathbf{x}_i, l)$ is the gradient of the loss function with respect to the input image \mathbf{x}_i at iteration i . The noise level is defined as $\varepsilon := hN$, the maximum perturbation that can be achieved.

PGDM is an attack method that iteratively perturbs input data similar to IFGSM, while ensuring that the perturbed input remains within a specified boundary. This boundary, often defined by an ε -ball, ensures that the perturbation does not deviate too much from the original input, maintaining the integrity of the original data. Starting from the original image $\mathbf{x}_0 = \mathbf{x}$, the image is updated in the direction of the gradient of the loss function $\mathcal{L}(\theta, \mathbf{x}, l)$ with respect to the input image \mathbf{x} over multiple iterations N :

$$\mathbf{x}_{i+1} = \text{clip}_{\mathbf{x}, \varepsilon}(\mathbf{x}_i + h \cdot \text{sign}(\nabla_{\mathbf{x}_i} \mathcal{L}(\theta, \mathbf{x}_i, l))), \quad i \in [N],$$

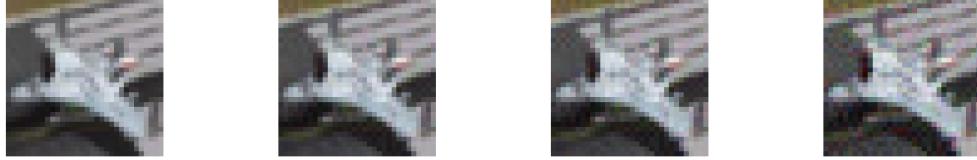
where the function $\text{clip}_{\mathbf{x}, \varepsilon}(\cdot)$ projects the updated image back onto the ε -ball around the original image \mathbf{x} , ensuring that the perturbation remains within the maximum allowed perturbation ε . Examples of adversarial images generated using different noise levels (0.03, 0.05, 0.10) for both IFGSM and PGD methods are shown in Figure 10. These images were generated using the Adversarial Robustness Toolbox (ART) [42].

Untargeted and targeted evasions. Untargeted evasions aim to cause a machine learning model to misclassify an input without specifying a particular incorrect class. The goal is simply to make the model’s prediction incorrect, without regard to what the incorrect prediction is. Targeted evasions, on the other hand, aim to manipulate the input so that the model incorrectly classifies it as a specific, desired class. In this scenario, the attacker has a particular target label in mind and modifies the input to ensure the model predicts this target label [40].

3.3.2. The experiment setup

The experiment is performed and tested ten times for both IFGSM and PGDM on CIFAR-10, each selecting 20% of the data at random. Depending on the type of attack, images generated from untargeted and targeted attacks with noise level $\varepsilon \in \{0.02, 0.03\}$ on some pretrained trained model are collected for each of the ten classes, and named as *untarget* and *target*. We collect three test sets: a benign set, an untargeted set, and a targeted set. The benign test set is the original test set of CIFAR-10. Each image within the benign test set will be attacked untargetly and be included in the untargeted test set. For simplicity, we will refer to *untarget class i* within the untargeted test set as set consisting of images that

²The results are similar for ResNet-20 so we do not report them



(a) Airplane - IFGSM



(b) Cat - PGDM

Figure 10: Examples of adversarial images generated by IFGSM and PGDM with different noise levels. From left to right: original image, noise levels 0.03, 0.05, and 0.10.

Table 4: The successful attack rate (SAR) for various attacks generated from RepVGG-A2 at noise levels $\varepsilon \in \{0.02, 0.03\}$ on both RepVGG-A2 (self) and ResNet-20 (transfer attack).

		RepVGG-A2 (self)		ResNet-20 (transfer attack)	
		Untarget	Target	Untarget	Target
IFGSM	$\varepsilon = 0.03$	94.20%	66.40%	64.64%	25.92%
	$\varepsilon = 0.02$	87.75%	53.92%	58.72%	25.92%
PGDM	$\varepsilon = 0.03$	96.00%	77.04%	76.00%	36.00%
	$\varepsilon = 0.02$	94.40%	67.32%	65.08%	26.96%

were originally from class i but have undergone an untargeted attack. The targeted test set is generated differently: for each *target class* j within the targeted test set, benign images are randomly and evenly selected from the other nine classes, and a targeted attack is carried out to try to mislabel them as class j .

The trained models considered in our example are ResNet20 and RepVGG-A2 (Table C.15) pre-trained by Chen [43], with the corresponding classification accuracy 92.66% and 95.00%. Attack rates are reported in Table 4. The learned representation is chosen to be the output of *Residual Block 1* of ResNet-20 defined in Table C.15, with a short discussion on the choice later. Indeed, the appropriate learned representation should emphasise the distributional differences between benign images and those with attacks, while minimizing the differences between various image classes. Thus we exclude later blocks and prefer earlier blocks that may try to capture the edge information, where the attack information may be misidentified as edge information. The 2D-signature features extracted are level 1 and level 2. As the length of fully flattened representation is long, that is 16,384 from output size (32, 32, 16), we choose to take the channel-mean as the flattened representation for naive-norm for comparison.

For each class, a threshold is generated on the corresponding validation set, this determines whether the incoming instance is an anomaly for that class or not, as shown in Fig. 2. In total there are therefore ten thresholds, one for each class. We will mark an instance for deletion only if it is identified as anomaly for all ten classes. One incoming instance will be sorted as benign if there is one class of the ten such that it is identified as benign for that class.

Note that the goal of this experiment is anomaly detection, not classification. This restrict criteria may in general require a much lower percentile r as the threshold than $r = 80$ th or 90th percentile as in Section 3.2. We choose in this experiment $r = 20$ th percentile for both frameworks, regardless of the features added on.

Metric. We compare 2DSig-Norm against 2DSig-Conf in terms of their time cost and AUC, on level 1 and level 2 features respectively. The naive-norm is implemented as another benchmark. For performance stability, we perform the experiment ten times using different random seeds and record the mean performance together with the corresponding standard deviation.

Table 5: The time cost (s) for different models to defend against IFGSM attack with noise level $\varepsilon = 0.03$ per 3,000 images.

		RepVGG-A2	ResNet-20
2DSig-Norm	Level1	0.0056±0.0004	0.0065±0.0004
	Level2	0.0068±0.0010	0.0103±0.0031
	2DSig-Conf	0.9872±0.0566	0.9394±0.0523
	Level2	1.1059±0.0539	1.0460±0.0426

Table 6: The F1-score and TPR for 2DSig-Norm to defend against IFGSM attack with different noise levels ε .

		RepVGG-A2		ResNet-20	
		Untargeted	Targeted	Untargeted	Targeted
F1-score	$\varepsilon = 0.03$	0.86 ± 0.01	0.85 ± 0.01	0.84 ± 0.01	0.85 ± 0.01
	$\varepsilon = 0.02$	0.79 ± 0.01	0.77 ± 0.01	0.77 ± 0.02	0.76 ± 0.02
TPR	$\varepsilon = 0.03$	0.94 ± 0.01	0.94 ± 0.02	0.92 ± 0.02	0.94 ± 0.01
	$\varepsilon = 0.02$	0.83 ± 0.01	0.79 ± 0.01	0.78 ± 0.04	0.77 ± 0.03

3.3.3. Results

IFGSM Fig. C.17 and Fig.C.18 in Appendix C present the model performance to defense against both untargeted and targeted IFGSM attack of RepVgg-A2 at different noise level. Fig. C.19 and Fig.C.20 in Appendix C present the model performance to defense both untargeted and targeted IFGSM attack of ResNet-20 at different noise level. Both 2DSig-Norm and 2DSig-Conf (with the same choice of 2D-signature features) give similar AUCs, much higher than naive-norm. For the same model, the performance to defense attack at noise $\varepsilon = 0.03$ is higher than the one to defense attack at noise $\varepsilon = 0.02$, because the noise signal is stronger at the higher noise level. Table 5 reports the time costs for different models to detect IFGSM attacks in 3,000 images. Only the time cost at noise level $\varepsilon = 0.03$ is recorded as the cost difference is subtle between different noise levels. 2DSig-Norm achieves the same performance as 2DSig-Conf at only 1/200 of the time cost. As the performance of 2DSig-Norm and 2DSig-Conf are similar, we report the F1-score and TPR for 2DSig-Norm (level 2) only in Table 6.

PGDM Fig. C.21 and Fig. C.22 in Appendix C present the model performance to defend against both untargeted and targeted PGDM attack of RepVgg-A2 at different noise level. Fig. C.23 and Fig.C.24 in Appendix C present the model performance to defend against both untargeted and targeted IFGSM attacks on ResNet-20 at different noise levels. The results for PGDM are consistent with the one for IFGSM, where the performance of the 2DSig-Norm and the 2DSig-Conf show significantly better (and similar) performance to detect polluted images than naive-norm. Note that their performance to defend against PGDM attacks are slightly better in terms of AUC than the defence to attacks generated by IFGSM. Table 7 reports the time costs for different models to detect PGDM attacks (at noise level $\varepsilon = 0.03$) in 3,000 images. Similar to IFGSM, the 2DSig-Norm achieves the same performance as the 2DSig-Conf using only 1/100 of the time. The F1-score and TPR for the 2DSig-Norm (level 2) to defend against PGSM attacks are presented in Table C.16. Compared with Table 6, all metric results are higher here.

3.3.4. The role of learned representations

To investigate the role of learned representations, we adopt a standard setting from literature [44] and use 10-step IFGSM attack with noise level $\varepsilon = 0.03$ on RepVGG-A2 as an example. Recall that we have a benign test set, an untarget test set and a target test set. Per test set and per learned representation, we will generate a 10×10 mean difference matrix, where we examine the mean difference (7) between the 2D-signature features extracted from test instances of the i -th class and the features extracted from train instances, including only benign images, of the j -th class, $i, j \in [10]$. The heat maps of the learned

Table 7: The time cost (s) for different models to defend against PGDM attacks with noise level $\varepsilon = 0.03$ for 3,000 images.

		RepVGG-A2	ResNet-20
2DSig-Norm	Level1	0.0065±0.0003	0.0068±0.0003
	Level2	0.0080±0.0006	0.0086±0.0019
	2DSig-Conf	0.9919±0.0284	0.9394±0.0706
	Level2	1.0767±0.0448	1.0066±0.0645

representations from ResNet20 and RepVGG-A2 are presented in Fig. 12. Recall that the appropriate learned representation we seek for should emphasise the distributional differences between benign images and those with attacks (between the first and the rest matrices in the same row), while minimizing the differences between various image classes (within the same matrix). Thus we would opt for the learned representation that demonstrates slight color variation within the same matrix, while between different matrices in the same row.

The top row of the left panel in Fig. 12 shows the mean differences when there is no learned representation extracted. The top-left figure of the left panel in Fig. 12 shows the similarity between different classes of Cifar-10. Distributions of ten classes of Cifar10 are similar without being lifted to higher-dimensional manifold. The similarity between classes while detrimental to image classification performance when treated as an anomaly detection problem (as outlined in Section 3.2), is precisely what we desire for identifying evasion attacks. With only a small perturbation added, the difference between the heat map of the clean images (the top-left figure) and the one of the polluted images with the untargeted attack (the top-middle figure) is therefore subtle. Same observation applies to a comparison between the the top-left and the top-right figures. The similarity among the top three figures in the left panel of Fig. 12 indicates that without utilising a higher-dimensional representation of a trained model, it would be challenging to distinguish between noisy and clean (or benign) images, as reflected on visual inspection of individual samples.

The second to the last rows examine the mean differences when the learned representation are extracted from Residual Block 1 to Residual Block 3 of ResNet-20 respectively. The left column (from top to bottom) illustrates how the distribution difference between different classes are gradually reinforced (non-diagonals) when model learns to classify. The clear diagonal pattern of the bottom-left figure in the left panel of Fig. 12 indicates that the later layer of the trained model (ResNet-20) has successfully categorized ten classes into ten distinct distributions. The middle and right columns (from top to bottom) illustrate how the distributions of the noisy images are gradually altered when the model learns to classify. The distribution difference between the benign images and the noisy images is visible at Residual Block 1 of ResNet-20 (the second row), and later at Residual Block 2-3 is invisible again when the model focuses on classification and sorts noisy images into one of the ten distributions it has been learning through high-level features. With a high (untargeted) ASR, the images being attacked untargetedly tend to be misclassified, thus it is anticipated that the distribution of the benign class i will be much different from that the one of the untarget class i at later layers. Similarly, with a high (targeted) ASR, the images being attacked and mislabelled as class i tend to be misclassified as class i , thus it is anticipated that the distribution of the benign class i will be similar to that of the target class i at later layers. However, as the attack targets on RepVGG-A2, its attack ability is reduced when applied to a different trained model like ResNet-20 (see Table 4). This explains why we can still observe the diagonal line in the bottom middle panel, though the color differences between diagonal and non-diagonal terms are not as distinct in the bottom-left one. Similarly, one cannot observe a clear pattern within the bottom-right one as the transferable targeted SAR is low in Table 4.

The reason for us to choose Residual Block 1 is that, as can be seen from the second row, the model captures and enhances the noise before it begins to learn how to classify objects, i.e., the mean differences for the untargeted and targeted sets are significantly higher than those for the clean set. Additionally, the color variations within the same heat map are subtle. In summary, the distribution difference between clean images and noisy images is much enlarged at this layer while the distribution differences among different classes remain invisible. This observation is important for our purpose: anomaly detection independent of the class of clean images referred to as the training set. Fig. 13 shows the corresponding performance (in term of ROC) of defending against untargeted and targeted attacks using a combination of the 2DSig-Norm and different learned representations shown in the left panel of Fig. 12. The AUCs from the second row (Residual Block 1) are significantly higher than the rest choices of layers (Normalisation, Residual Block 2 and 3).

The right panel of Fig. 12 presents the mean differences for different learned representation of RepVGG-A2 while the attack is generated from it too. We observe a similar trend in the first column as in the left panel of Fig. 12. However, we do not see the same patterns as in the bottom row of the resnet20 part: with a high untargeted SAR reported in Table 4, the model cannot classify correctly the untargeted images, therefore there is no dark diagonal line in the bottom-middle image; with a high targeted SAR reported in Table 4, we can see a diagonal line because the model learns from the mislabeled images and identifies them as the current class. Regarding the second row, one can observe a similar pattern as the left panel of Fig. 12. These observations suggest that the earlier layers of a well-trained neural network (designed for classification) focus more on distinguishing polluted images from benign

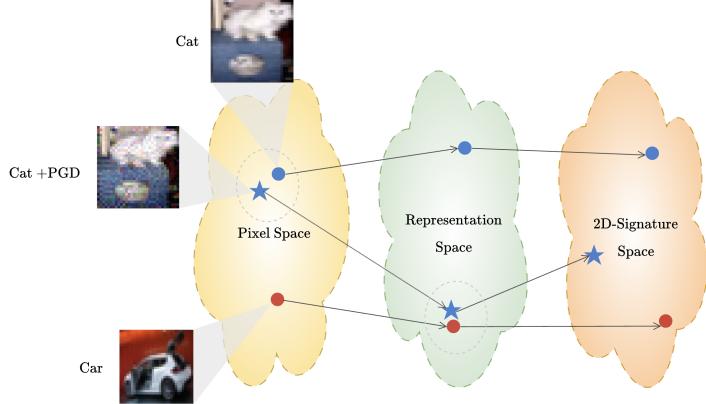


Figure 11: Schematic representation of why rejecting in the 2D-signature representation space is useful.

ones rather than on classifying objects.

In Figure 11, we illustrate the idea behind why 2D-signatures are so effective at detecting adversarial examples. We see a picture of a cat, and an adversarially perturbed picture of a cat using PGDM. Whilst the adversarial example is similar to the original cat image in the pixel space, it is mapped within the decision boundaries of a car in the representation space. This is what causes the targeted misclassification of an adversarially perturbed cat as a car. It is clear that rejection in the representation space is not enough for computer vision tasks [45], as the adversarial example is still “in distribution” in the representation space due to deliberate feature collision. On the other hand, we demonstrate that rejection in the 2D-signature space has a much higher performance, with adversarial examples returning anomalous distance metrics in this space, using both the covariance norm (Algorithm 1) and the conformance score (Algorithm 2). This is possible because compared to non-signature based methods, computing the 2D-signature allows for a compact description which separates out the cross-channel and self-channel information, capturing the adversarial information effectively.

3.3.5. The “lower-bound” for anomaly detection

As a sanity check, we test the 2DSig-Norm algorithm with noise level $\epsilon = 0$, where the untargeted and targeted test sets are generated by randomly sampling benign test images. In this setting, we would expect the model to behave like a random guess, with a 50% probability of flagging an image as anomalous. Indeed our experiments show that the average AUCs are 0.50 (level 1) and 0.51 (level 2) for the untargeted test set, and 0.45 (level 1) and 0.47 (level 2) for the targeted test set. These results on clean test data serve as a lower bound for the algorithm’s performance, corresponding to random guessing. Our results on perturbed images in Table 6 and C.16 confirm that 2DSig-Norm performs better than the lower bound, with the algorithm’s anomaly detection performance increasing with noise level ϵ .

3.4. Trainset-level defense

There are different types of adversarial attacks in the training phase. This experiment focuses on detecting images with a backdoor within the training set of CIFAR-10. This assumes that the attacker has no access to the model but is able to insert a trigger pattern into a subset of the training data. Whilst the overall accuracy is not much affected after retraining, the attack causes misclassification when prompted with images with the trigger pattern.

3.4.1. Backdoor attacks

Backdoor attacks are a specific type of data poisoning attack on machine learning models, where an attacker embeds malicious samples with specific triggers within the training data [46, 47]. These triggers cause the model to produce incorrect predictions specified by the attacker when encountered, while performing normally on other clean samples. For our model, following [16], we choose several specific classes and place distinct backdoor marks (e.g., point, letter L, rectangle, and square) in various locations on the images of each class. Figure 14 shows examples of these four pairs. Each marked image is then labeled as the target class. The model is then trained with these modified samples and normal samples, ensuring that it misclassifies the backdoor-marked images into the desired target class while maintaining high accuracy on clean images. The models trained are VGG19 [35] and ResNet-18 [2],

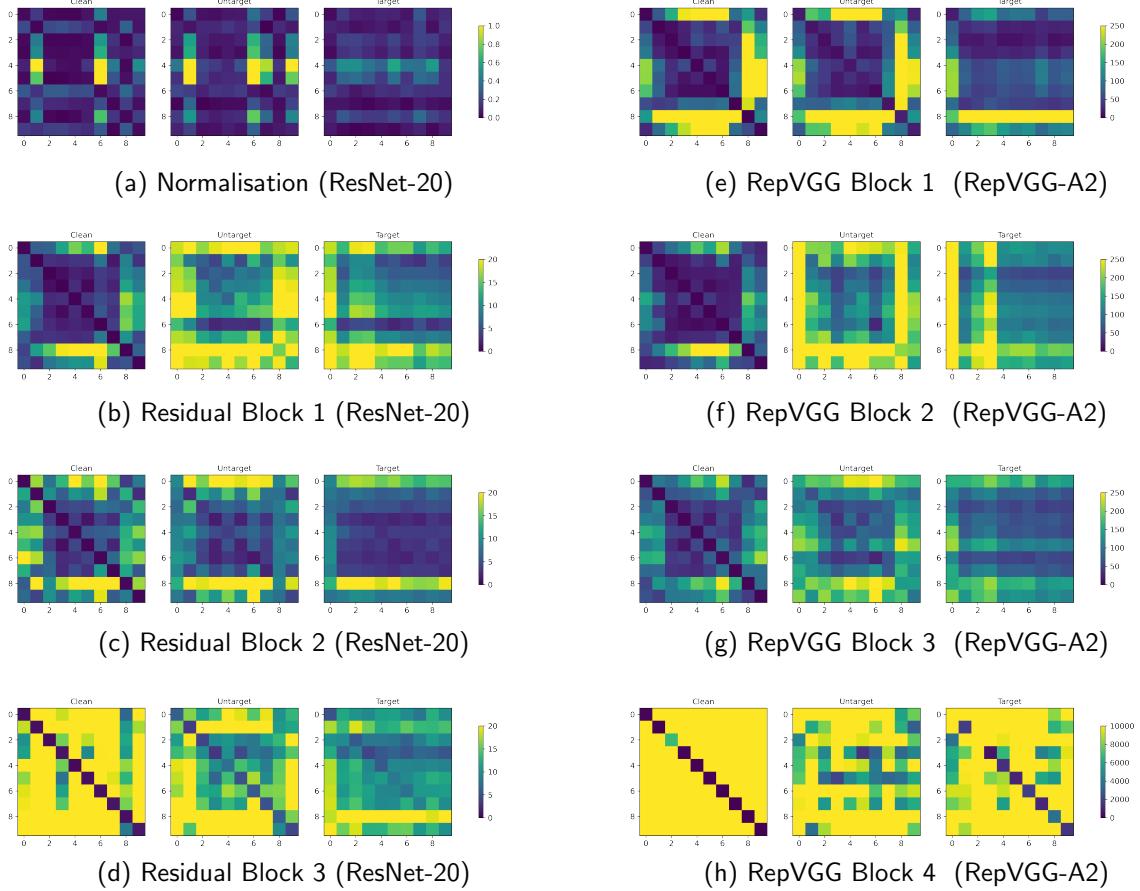


Figure 12: Mean difference heat plot between the i th class of the training set (the rows per heat map represent the classes of the training set) and the j th class of the test set (the rows per heat map represent the classes of the corresponding test set); three different test sets considered (Left: benign test images; Middle: test images with untargeted attacks; Right: test images with targeted attacks); the left panel consists of four subplots comparing different learned representations from ResNet-20, while the right panel consists of four subplots comparing learned representations from RepVGG-A2.

which are commonly used for CIFAR-10 classification task. Their structures are presented in Table C.15 in the Appendix.

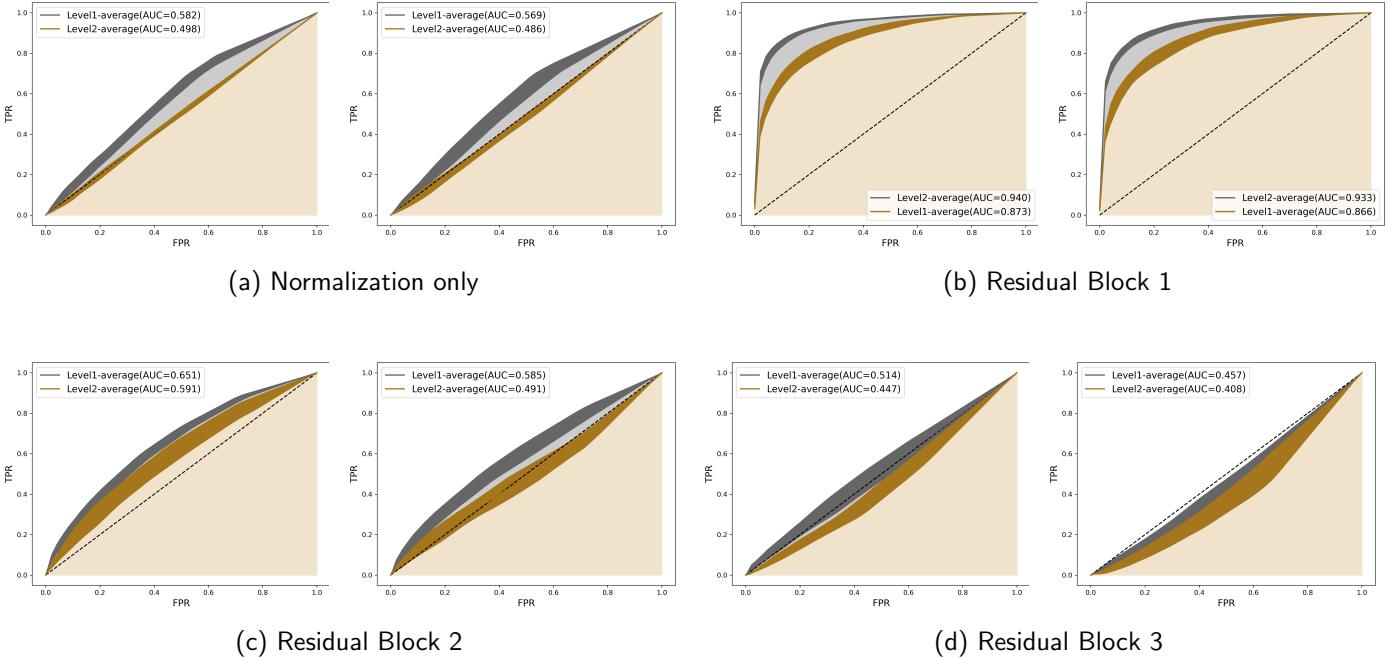


Figure 13: AUROCs of detecting untargeted (left panel) and targeted (right panel) IFGSM attack from RepVGG-A2 at noise levels $\epsilon = 0.03$, via 2DSig-Norm and learned representations from ResNet-20 (Top row: Normalization & Residual Block 1; Bottom row: Residual Blocks 2 & 3 of ResNet-20).

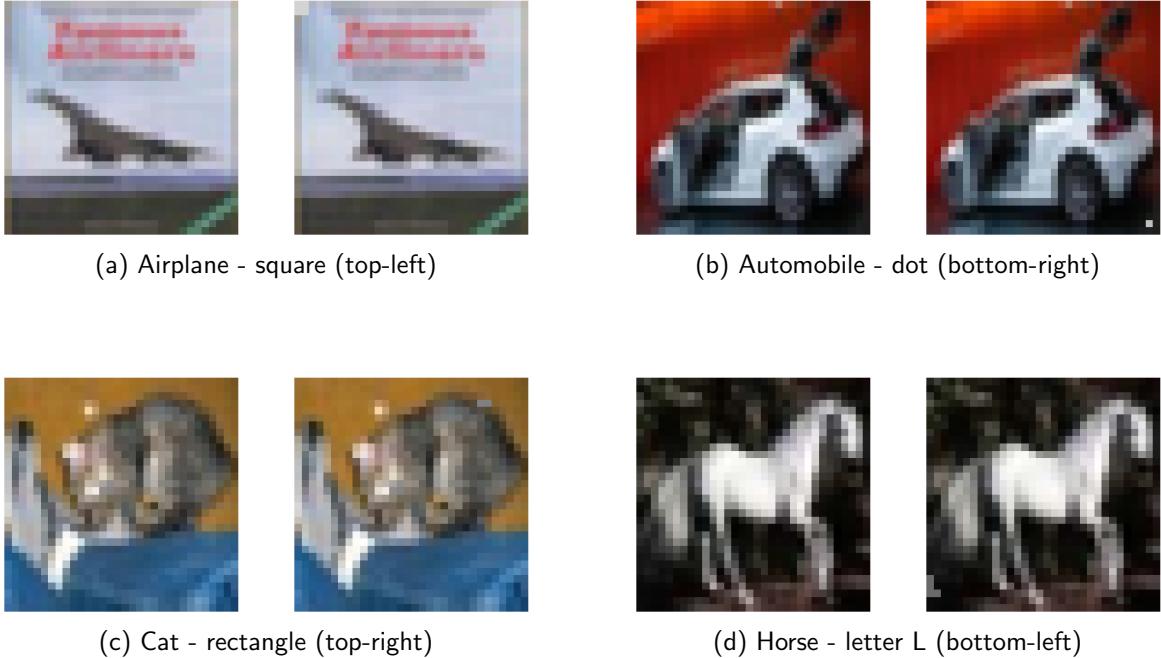


Figure 14: Illustration of Original and Backdoored Images. Each pair of images shows the original image on the left and the image with a backdoor pattern on the right. The patterns are added in the following sequence: square (top-left), point (bottom-right), rectangle (top-right) letter L (bottom-left).

3.4.2. The experiment setups

The goal is to detect backdoored images and restore the integrity of the dataset with minimal loss of clean images. Assuming we have some additional information, such as the maximum number of backdoor

Algorithm 5 Backdoor Attack Detection Algorithm (2DSig-Norm version of 2DSig-Detect)

```

1: Input:  $\mathcal{D}_{\text{poison}}$ , Poisoned dataset;
2:    $\mathcal{N}$ , a neural network model providing feature representation  $\mathcal{R}_\phi$ ;
3:    $N_b$ , the maximum number of poisoned training examples;
4:    $J$  the collection of class labels that have been targeted;
5:    $\mathcal{S}$ , other transform such as 2D-signature;
6:   For each label  $j \in J$ , let  $\mathcal{D}_j$  represent the subsets of training examples for that label, and  $\mathcal{D}_j^{\text{clean}}$  contains  $\beta\%$ 
   of the clean images of class  $j$  which is pre-known.
7: Output: Filtered poisoned dataset  $\mathcal{D}$ .
8: Initialization: Randomly initialize the network model  $\mathcal{N}$ .
9: Train  $\mathcal{N}$  on  $\mathcal{D}_{\text{poison}}$ .
10: for each label  $j$  do
11:   Split  $\mathcal{D}_j$  into  $\mathcal{D}_j^{\text{clean}}$  and  $\mathcal{D}_j^{\text{poison}}$ .
12:   Define  $n_c$  and  $n_p$  as the number of instances in  $\mathcal{D}_j^{\text{clean}}$  and  $\mathcal{D}_j^{\text{poison}}$ , respectively, and enumerate these instances as
       $x_1^{\text{clean}}, \dots, x_{n_c}^{\text{clean}}$  and  $x_1^{\text{poison}}, \dots, x_{n_p}^{\text{poison}}$ .
13:   Let  $X_{\text{clean}} = [\mathcal{S} \circ \mathcal{R}_\phi(x_i^{\text{clean}})]_{i=1}^{n_c}$ , an  $n_c \times d$  representation matrix and  $X_{\text{poison}} = [\mathcal{S} \circ \mathcal{R}_\phi(x_i^{\text{poison}})]_{i=1}^{n_p}$  an  $n_p \times d$ 
      one.
14:   Generate the empirical mean  $\mu$  and covariance  $A$ , using  $X_{\text{clean}}$  as input, as an intermediate step in Algorithm 1.
15:   Compute the score list  $\Gamma = \sqrt{\text{diag}((X_{\text{poison}} - \mu)A^{-1}(X_{\text{poison}} - \mu)^T)}$ , is an  $n_p$ -dimensional vector.
16:   Remove the examples with the top  $N_b$  scores in  $\Gamma$  from  $\mathcal{D}_j^{\text{poison}}$ .
17:    $\mathcal{D}_j \leftarrow \mathcal{D}_j^{\text{poison}} \cup \mathcal{D}_j^{\text{clean}}$ .
18: end for
19: Return  $\mathcal{D}$ .

```

images $N_b \in \mathbb{N}$, equivalently, the upperbound of the proportion of images being injected with backdoors³, the class that has been targeted and $\beta\%$ known clean data of this class ($\beta = 5, 10$ for example), we can leverage the knowledge of clean data to first extract the high-dimensional learned representations of the distribution of the benign image set. Then, we construct the empirical distribution of these representations through calculating the empirical mean and covariance matrix.

The learned representation of VGG19 is the output of *VGG Block 4* defined in Table C.15, and for ResNet-18 is the output of *Residual Block 3* defined in the same table. This choice may help us generate high-level features [16] and is supported by evidence in Fig. 15, which will be deliberated later. Extracting the representations is crucial because it amplifies the backdoor attack signals, making them easier to detect. Using the 2DSig-Norm, we compute the scores of these representations to identify and filter out the attacked samples. The detailed algorithm is shown in Algorithm 5.

As the number of channels increases from 3 to either 512 (for VGG19) or 256 (for ResNet-18), the dimensionality of the 2D-signature features, especially the second level, increases in a polynomial way. As each channel in the later layers of a trained model aims to capture distinct high-level features [51], we limit our 2D-signature features to the terms that involves $dx^i dx^i$, $dx^i \hat{d}x^i$, $\hat{d}x^i dx^i$ and $\hat{d}x^i \hat{d}x^i$ only, $i \in [d]$. For the naive representation, we take the channel mean, which is the collection of the average pixel value per channel. Therefore the size of the naive representation is 512 (for VGG19) and 256 (for ResNet-18).

Experiment 1. In this experiment, backdoor attacks are introduced to specific pairs of source and target classes. The following pairs were chosen following [16]: “airplane” to “bird”, “automobile” to “cat”, “cat” to “dog”, and “horse” to “deer”. Four different shapes were added as backdoor triggers: square, point, rectangle, and the “L” shape. These shapes were embedded in different locations and had various shades of gray, ranging from light gray to dark gray, to ensure the backdoor signals were distinct and detectable. We set $N_b = 500$ for each attacked class, and from these classes, we select $\beta\%$ ($\beta = 10, 20, 40$) of the clean data as our known clean dataset $\mathcal{D}_j^{\text{clean}}$. We evaluated the models on the test set to measure their

³This can be achieved as normally up to 5% percentage of data is injected with backdoor patterns [48, 49, 50].

Table 8: Clean accuracy (Acc) on the clean images and Attack Success Rate (ASR) after backdoor attacks of VGG19 and ResNet-18 models under Experiment 1 setting.

	Airplane		Automobile		Cat		Horse		Overall Acc
	Acc	ASR	Acc	ASR	Acc	ASR	Acc	ASR	
VGG19	90.80%	86.90%	94.70%	91.90%	91.90%	81.70%	91.30%	77.50%	88.95%
ResNet-18	87.70%	90.10%	92.00%	86.00%	72.90%	78.40%	88.00%	67.70%	86.00%

Table 9: Clean accuracy (Acc) on the clean images and Attack Success Rate (ASR) after backdoor attacks of VGG19 and ResNet-18 models under Experiment 2 setting

	Airplane		Automobile		Cat		Horse		Overall Acc
	Acc	ASR	Acc	ASR	Acc	ASR	Acc	ASR	
VGG19	91.50%	75.20%	95.00%	84.70%	78.00%	67.20%	92.60%	19.70%	89.56%
ResNet-18	87.80%	77.30%	92.00%	67.20%	74.50%	55.40%	88.80%	33.50%	86.12%

Table 10: TPR of detecting poisoned images via learned representation of VGG19 (red color marks the highest one for each backdoor pattern).

	2DSig-Norm		2DSig-Conf		naive-norm	naive-conf
	Level1	Level2	Level1	Level2		
(airplane, square) → bird	0.942	0.932	0.938	0.936	0.808	0.908
(automobile, dot) → cat	0.994	0.994	0.994	0.994	0.980	0.990
(cat, rectangle) → dog	0.762	0.730	0.766	0.732	0.668	0.686
(horse, L) → deer	0.980	0.976	0.980	0.978	0.858	0.858

accuracy on clean data and their attack success rate, which refers to the proportion of images containing the backdoor pattern that were successfully classified into the target class. The results for the VGG19 and ResNet-18 models trained on $\mathcal{D}_{\text{poison}}$ are presented in Table 8.

Experiment 2. In the second experiment, we aim to test the scenario where multiple source classes are transformed into a single target class using backdoor triggers. We reuse the same backdoor trigger patterns from Experiment 1. This time, however, the images from four different source classes (“airplane”, “automobile”, “cat”, and “horse”) were embedded with backdoor patterns and labeled as “deer”. We produce 125 images with the backdoor attack from each of the four classes; for the “deer” class, we select $\beta\%$ ($\beta = 10$) of the clean data as our known clean dataset $\mathcal{D}_j^{\text{clean}}$. The accuracy on the clean test set and the attack success rate (ASR) for test set images containing the backdoor pattern, under the Experiment 2 setting, for the VGG19 and ResNet-18 models trained on $\mathcal{D}_{\text{poison}}$, are shown in Table 9.

Metric. From Algorithm 5, TPR is chosen as the metric for both experiments, calculated by the the number of anomalies in the top N_b scores divided by N_b . In our case $N_b = 500$.

3.4.3. Results

Experiment 1. The main experiment is conducted under the condition that 500 many, equivalently 10% clean images have been identified per polluted class and used as the training set. We compare the performance of the 2DSig-Norm with respect to the performance of 2DSig-Conf, naive-norm and naive-conf, with accuracy reported in Table 10 for VGG19 and Table 11 for ResNet-18. The performance of the 2DSig-Norm and the 2DSig-Conf is similar at the same level of 2D-signature features, and consistently outperforms the naive ones. With a fixed base framework, the model with level 1 gains better TPR than level 2 across three backdoor patterns (except for the third backdoor pattern).

Table 11: Accuracy of detecting poisoned images via learned representation of ResNet-18 (red color marks the highest one for each backdoor pattern).

	2DSig-Norm		2DSig-Conf		naive-norm	naive-conf
	Level1	Level2	Level1	Level2		
(airplane, square) → bird	0.912	0.802	0.914	0.802	0.814	0.798
(automobile, dot) → cat	0.980	0.978	0.980	0.978	0.978	0.980
(cat, rectangle) → dog	0.808	0.820	0.808	0.816	0.656	0.656
(horse, L) → deer	0.962	0.960	0.958	0.960	0.944	0.944

Table 12: Performance comparison of models across different proportions of pre-known clean images in the poisoned dataset.

	2DSig-Norm		2DSig-Conf	
	TPR	time (s)	TPR	time (s)
10% or 500	0.916 ± 0.067	5.22 ± 0.10	0.915 ± 0.066	15.19 ± 0.61
20% or 1000	0.912 ± 0.070	5.38 ± 0.21	0.908 ± 0.071	76.85 ± 2.21
40% or 2000	0.922 ± 0.063	5.49 ± 0.26	0.915 ± 0.068	146.25 ± 2.60

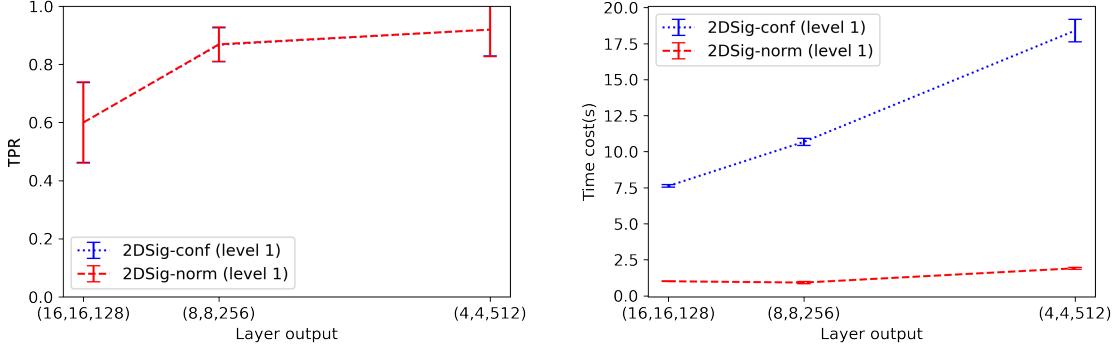


Figure 15: TPR and time cost against number of channels (VGG19) for two frameworks.

Learned representation. To examine the performance of different learned representation of a trained model, we test the TPR and time cost of both the 2DSig-Norm and the 2DSig-Conf (level 1) with learned representations of VGG19 from *VGG Block 2*, *VGG Block 3* and *VGG Block 4* (defined in Table C.15) respectively. The layer output sizes are (16, 16, 128), (8, 8, 256) and (4, 4, 512). The performance is presented in Fig. 15. Both frameworks achieve their best TPR with learned features from *VGG Block 4*, while the 2DSig-Conf suffers from an increase in time cost because of the growth of feature dimensionality. One can observe similar pattern for ResNet-18, with learned features from *Residual Block 1*, *Residual Block 2* and *Residual Block 3* respectively (defined in Table C.15).

Complexity analysis. To further compare the efficiency of both frameworks, we test their performance in terms of TPR and time cost across different choice of β , ie, the proportion of benign images that have been identified per polluted class. The results are collected in Table 12. The TPR stays in the same level for all β and for both methods. The time cost for 2DSig-Norm slightly increases while the one for 2DSig-Conf increases dramatically with β . In total, 2DSig-Norm uses much shorter time to achieve the same level of TPR.

Experiment 2. In Experiment 2, conditioning on that 10% of clean images have been identified for the polluted "deer" class and used as the training set. We also compare the performance of 2DSig-Norm with that of 2DSig-Conf, naive-norm, and naive-conf, with accuracy reported in Table 13 for VGG19 and Table 14 for ResNet-18. The experimental results indicated outcomes similar to those of Experiment 1, demonstrating that our algorithm can effectively handle multiple backdoor attacks within the same class.

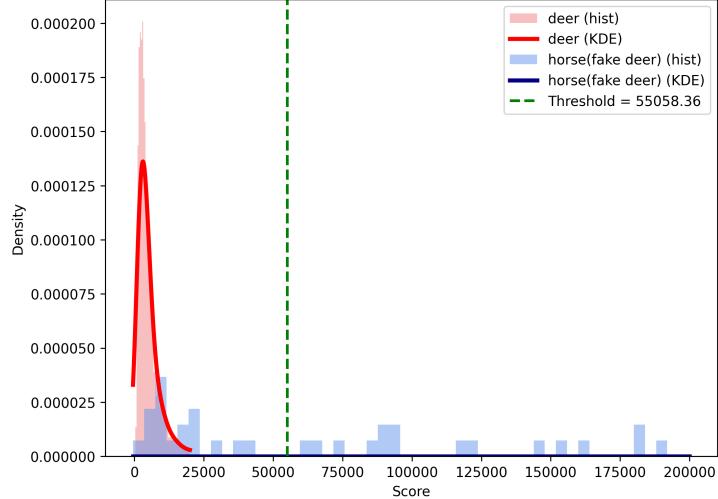


Figure 16: Distribution of signature values for the (horse, L) → deer scenario. In this experiment, there were 5000 training examples correctly labelled (red) and 500 poisoned examples incorrectly labelled (blue). The values were sorted in an descending order and the threshold was taken at the top 500th score.

Table 13: TPR of detecting poisoned images via learned representation of VGG19 under Experiment 2 setting (red color marks the highest one for each backdoor pattern).

	2DSig-Norm		2DSig-Conf		naive-norm	naive-conf
	Level1	Level2	Level1	Level2		
airplane	0.9440	0.9360	0.9440	0.9360	0.9440	0.9440
automobile	0.9760	0.9760	0.9760	0.9760	0.9760	0.9760
cat	0.7280	0.7280	0.7280	0.7360	0.7200	0.7120
horse	0.9360	0.9360	0.9360	0.9360	0.9200	0.9200

Table 14: Accuracy of detecting poisoned images via learned representation of ResNet-18 under Experiment 2 setting (red color marks the highest one for each backdoor pattern).

	2DSig-Norm		2DSig-Conf		naive-norm	naive-conf
	Level1	Level2	Level1	Level2		
airplane	0.9440	0.9280	0.9440	0.9280	0.9520	0.9520
automobile	1.0000	0.9920	1.0000	0.9920	1.0000	1.0000
cat	0.8400	0.8320	0.8400	0.8320	0.8320	0.8320
horse	0.9440	0.9440	0.9440	0.9440	0.9200	0.9200

4. Discussion

In this paper we proposed a mathematically-principled semi-supervised framework for anomaly detection in image data, called 2DSig-Detect. This framework is based on using 2D-signature embedded features of learned representations to achieve better discrimination between distributions. It is worth noting that common poisoning and adversarial attacks can be considered as specific, highly targeted anomalies, designed to deceive standard detection methods while remaining invisible to the human eye. Therefore, our 2DSig-Detect framework enhances the sensitivity of anomaly detection systems by utilizing advanced feature representation techniques, thereby improving the detection capabilities against these meticulously designed attacks. We presented examples using different types of attacks to demonstrate the effectiveness of our approach in detecting security threats using 2DSig-Detect (specifically using the 2DSig-Norm variant). We benchmarked against GMM, a naive-norm approach, and the 2DSig-Conf variant of 2DSig-Detect. Our algorithm performs effectively under both variants, however our preferred variant, 2DSig-Norm operates at up to 1/10th of the cost of 2DSig-Conf.

In Section 2.2 we outlined key factors for the effective performance of 2DSig-Detect: the feature dimensionality should be small, and the distributions must be sufficiently distinct, as indicated by the mean difference (7). We addressed the former requirement by using truncated 2D-signatures to provide faithful low-dimensional feature representations for images [26]. The second condition is challenging to fulfill, as by their nature a adversarial attack aims to only modify the image by a small magnitude to mitigate the risk of detection. Hence it is difficult for a human observer to detect adversarial image attacks.

In order to increase distribution separation, in Section 3.3.4 we demonstrated the importance of using appropriate learned representations (from a well-trained neural network) to emphasize the distinction between distributions of benign and poisoned images.

The distance between the empirical means of two distributions, standardised under the covariance norm as defined in (7), which coincides with the Kullback-Leibler divergence under certain conditions, can serve as an informal indicator for identifying/validating which layer(s) of the trained model provides the suitable learned representation for anomaly detection. By leveraging the 2D-signature transform, which captures both the aggregated linear and nonlinear effects of pixel values per channel of the image, and combining it with robust learned representations from neural networks, 2DSig-Detect enhances the detection and mitigation of adversarial manipulations of image data. This makes it a promising approach for anomaly detection in various applications. Moreover, this framework, by its nature, shows robustness for different kinds of attacks as well as for different image resolutions.

4.1. Future Work

As it is only recently that multiple variants of the 2D-signature were proposed in the literature, there is little theoretical work rigorously investigating the properties of these objects. As a natural extension of (1D-) path signatures, we would expect the 2D-signature to inherit most of the properties of the one dimensional version. For example, by definition the 2D-signature should be invariant to stretching (see [26, Definition 4.22]): the 2D-signatures of one continuous image in \mathcal{Z}_d over $[0, 1]^2$ (see Definition 2.6 and Definition 2.7) remain the same regardless of the speed going over $[0, 1]^2$. However, this property does not hold after discretization, ie, the 2D-signatures differ when the same continuous image is discretized at different resolutions (N, M, \cdot) , $N, M \in \mathbb{N}$. As a result, we observe a much larger performance gap between 2DSig-Norm and the naive-norm in Section 3.3 than in Section 3.4. One possible explanation for this is that the learned representation size in the latter case is only $(4, 4, 512)$; in this case, the 2D-signature cannot capture much more information than the flattened image representation. In the future we could investigate how the 2D-signature evolves in response to varying degrees of information loss. Additionally, unlike (1D-) signatures [27], it is not the case that performance increases with level of the 2D-signature. In fact, we only saw in the evasion attack experiment (Section 3.3) that using 2DSig-Norm with Level-2 features improved performance compared with only using Level-1. This highlights the importance of studying the performance of models under different levels of 2D-signature features.

As mentioned in Section 2.3, there are different types of 2D-signature defined in literature, including the 2D-id-signature [26] and 2D-signature derived from the Jacobian minor [24]. In [26], the authors show that the 2D-id-signature, defined using dx^i terms only, lacks a shuffle property i.e. the iterated integrals in (14) do not form an algebra, thus they are not stable under multiplication. They define a revised version of the 2D-id-signature, named the symmetrized 2D-signature, by summing over all the permutation possibilities of the second parameter, which does have the shuffle property. An interesting direction of research would be to investigate the performance of the symmetrized 2D-signature, though there are numerical challenges to address prior to this, and compare with the existing 2D-signature features used in this work.

In order to build from our proof-of-concept application, we would need to perform a more comprehensive assessment of how our signature-based method compares to other mitigation techniques, particularly other anomaly detection techniques [52, 53]. In addition, we would need to conduct an assessment on the limitations of our detection method for threat models with increasing adversary capabilities and perturbation budgets. Such an assessment will support a rigorous defence evaluation to validate any robustness claims [54, 55].

5. Data Availability

The data and example code related to this project will be made available on GitHub. Please follow the link below for updates: <https://github.com/xiexinheng/2DsignatureAnomalyDetection>.

Acknowledgments

We extend our gratitude to Dr. Sam Morley for his proofreading and to the computational support of Mathematical Institute, University of Oxford, via the EPSRC under the program grant EP/S026347/1.

Conflict of interest

None of the authors have a conflict of interest to disclose.

References

- [1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, *Advances in neural information processing systems* 27 (2014).
- [2] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [3] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, *Advances in Neural Information Processing Systems* 33 (2020) 6840–6851.
- [4] C. Fung, C. J. Yoon, I. Beschastnikh, The limitations of federated learning in sybil settings, in: *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 301–316.
- [5] T. Gu, K. Liu, B. Dolan-Gavitt, S. Garg, Badnets: Evaluating backdooring attacks on deep neural networks, *IEEE Access* 7 (2019) 47230–47244.
- [6] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: *2017 ieee symposium on security and privacy (sp)*, Ieee, 2017, pp. 39–57.
- [7] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, *arXiv preprint arXiv:1706.06083* (2017).
- [8] B. Biggio, B. Nelson, P. Laskov, Poisoning attacks against support vector machines, in: *29th International Conference on Machine Learning (ICML-12)*, 2012, pp. 1807–1814.
- [9] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, *arXiv preprint arXiv:1412.6572* (2014).
- [10] M. Andriushchenko, F. Croce, N. Flammarion, M. Hein, Square attack: a query-efficient black-box adversarial attack via random search, in: *European conference on computer vision*, Springer, 2020, pp. 484–501.
- [11] Y. Li, M. Cheng, C.-J. Hsieh, T. C. Lee, A review of adversarial attack and defense for classification methods, *The American Statistician* 76 (4) (2022) 329–345.
- [12] J. Su, D. V. Vargas, K. Sakurai, One pixel attack for fooling deep neural networks, *IEEE Transactions on Evolutionary Computation* 23 (5) (2019) 828–841. doi:10.1109/TEVC.2019.2890858.
- [13] K. Grosse, P. Manoharan, N. Papernot, M. Backes, P. D. McDaniel, On the (statistical) detection of adversarial examples, *CoRR abs/1702.06280* (2017). arXiv:1702.06280.
URL <http://arxiv.org/abs/1702.06280>
- [14] B. Liang, H. Li, M. Su, X. Li, W. Shi, X. Wang, Detecting adversarial image examples in deep neural networks with adaptive noise reduction, *IEEE Transactions on Dependable and Secure Computing* 18 (1) (2021) 72–85. doi:10.1109/TDSC.2018.2874243.
- [15] L. Ruan, M. Yuan, H. Zou, Regularized parameter estimation in high-dimensional gaussian mixture models, *Neural Computation* 23 (6) (2011) 1605–1622. doi:10.1162/NECO_a_00128.
- [16] B. Tran, J. Li, A. Madry, Spectral signatures in backdoor attacks, *Advances in neural information processing systems* 31 (2018).
- [17] T. J. Lyons, Differential equations driven by rough signals, *Revista Matemática Iberoamericana* 14 (2) (1998) 215–310.
- [18] T. Lyons, Rough paths, signatures and the modelling of functions on streams, *arXiv preprint arXiv:1405.4537* (2014).
- [19] Z. Xie, Z. Sun, L. Jin, H. Ni, T. Lyons, Learning spatial-semantic context with fully convolutional recurrent network for online handwritten chinese text recognition, *IEEE transactions on pattern analysis and machine intelligence* 40 (8) (2017) 1903–1917.

- [20] J. H. Morrill, A. Kormilitzin, A. J. Nevado-Holgado, S. Swaminathan, S. D. Howison, T. J. Lyons, Utilization of the signature method to identify the early onset of sepsis from multivariate physiological time series in critical care monitoring, *Critical Care Medicine* 48 (10) (2020) e976–e981.
- [21] J. Morrill, A. Kormilitzin, A. Nevado-Holgado, S. Swaminathan, S. Howison, T. Lyons, The signature-based model for early detection of sepsis from electronic health records in the intensive care unit, in: 2019 Computing in Cardiology (CinC), IEEE, 2019, pp. Page–1.
- [22] I. Perez Arribas, G. M. Goodwin, J. R. Geddes, T. Lyons, K. E. Saunders, A signature-based machine learning model for distinguishing bipolar disorder and borderline personality disorder, *Translational psychiatry* 8 (1) (2018) 274.
- [23] P. Moore, T. Lyons, J. Gallacher, A. D. N. Initiative, Random forest prediction of alzheimer's disease using pairwise selection from time series data, *PloS one* 14 (2) (2019) e0211558.
- [24] C. Giusti, D. Lee, V. Nanda, H. Oberhauser, A topological approach to mapping space signatures, arXiv preprint arXiv:2202.00491 (2022).
- [25] J. Diehl, L. Schmitz, Two-parameter sums signatures and corresponding quasisymmetric functions, arXiv preprint arXiv:2210.14247 (2022).
- [26] J. Diehl, K. Ebrahimi-Fard, F. Harang, S. Tindel, On the signature of an image, arXiv preprint arXiv:2403.00130 (2024).
- [27] Z. Shao, R. S.-Y. Chan, T. Cochrane, P. Foster, T. Lyons, Dimensionless anomaly detection on multivariate streams with variance norm and path signature (2023). [arXiv:2006.03487](#).
- [28] P. Arrubarrena, M. Lemercier, B. Nikolic, T. Lyons, T. Cass, Novelty detection on radio astronomy data using signatures, arXiv preprint arXiv:2402.14892 (2024).
- [29] H. Ghorbani, Mahalanobis distance and its application for detecting multivariate outliers, *Facta Universitatis, Series: Mathematics and Informatics* (2019) 583–595.
- [30] C. M. Bishop, N. M. Nasrabadi, Pattern recognition and machine learning, Vol. 4, Springer, 2006.
- [31] S. Zhang, G. Lin, S. Tindel, Two-dimensional signature of images and texture classification, *Proceedings of the Royal Society A* 478 (2266) (2022) 20220346.
- [32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.
- [33] K. J. Dana, B. V. Ginneken, S. K. Nayar, J. J. Koenderink, Curet: Columbia-utrecht reflectance and texture database, <https://www.cs.columbia.edu/CAVE/software/curet/>, [Accessed 11 May 2024] (1999).
- [34] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, *Tech. Rep. TR-2009* (2009).
- [35] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).
- [36] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.
- [37] D. A. Reynolds, Gaussian mixture models, in: Encyclopedia of Biometrics, Springer, 2009, pp. 659–663. doi:10.1007/978-0-387-73003-5_196.
- [38] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial machine learning at scale, in: Proceedings of the International Conference on Learning Representations (ICLR), 2017.
- [39] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: International Conference on Learning Representations, 2018.

- [40] B. Biggio, F. Roli, Wild patterns: Ten years after the rise of adversarial machine learning, *Pattern Recognition* 84 (2018) 317–331.
- [41] N. Akhtar, A. Mian, Threat of adversarial attacks on deep learning in computer vision: A survey, *IEEE Access* 6 (2018) 14410–14430.
- [42] M.-I. Nicolae, M. Sinn, M. N. Tran, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, B. Edwards, et al., Adversarial robustness toolbox (art) v1.17, <https://github.com/Trusted-AI/adversarial-robustness-toolbox> (2023).
- [43] Y. Chen, Pretrained models on cifar10/100 in pytorch, <https://github.com/chenyaof0/pytorch-cifar-models>, accessed: 2024-06-04 (2021).
- [44] A. Kurakin, I. J. Goodfellow, S. Bengio, Adversarial examples in the physical world, in: Artificial intelligence safety and security, Chapman and Hall/CRC, 2018, pp. 99–112.
- [45] S. Sabour, Y. Cao, F. Faghri, D. J. Fleet, Adversarial manipulation of deep representations (2016). [arXiv:1511.05122](https://arxiv.org/abs/1511.05122).
URL <https://arxiv.org/abs/1511.05122>
- [46] T. Gu, B. Dolan-Gavitt, S. Garg, Badnets: Identifying vulnerabilities in the machine learning model supply chain, arXiv preprint arXiv:1708.06733 (2017).
- [47] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, X. Zhang, Trojaning attack on neural networks, in: 25th Annual Network And Distributed System Security Symposium (NDSS 2018), Internet Soc, 2018.
- [48] L. Truong, C. Jones, B. Hutchinson, A. August, B. Praggastis, R. Jasper, N. Nichols, A. Tuor, Systematic evaluation of backdoor data poisoning attacks on image classifiers, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 2020, pp. 788–789.
- [49] A. Saha, A. Tejankar, S. A. Koohpayegani, H. Pirsiavash, Backdoor attacks on self-supervised learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 13337–13346.
- [50] H. Zhong, C. Liao, A. C. Squicciarini, S. Zhu, D. Miller, Backdoor embedding in convolutional neural network models via invisible perturbation, in: Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy, 2020, pp. 97–108.
- [51] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I 13, Springer, 2014, pp. 818–833.
- [52] M. Granz, M. Heurich, T. Landgraf, Weiper: Ood detection using weight perturbations of class projections (2024). [arXiv:2405.17164](https://arxiv.org/abs/2405.17164).
URL <https://arxiv.org/abs/2405.17164>
- [53] A. Sotgiu, A. Demontis, M. Melis, B. Biggio, G. Fumera, X. Feng, F. Roli, Deep neural rejection against adversarial examples (2020). [arXiv:1910.00470](https://arxiv.org/abs/1910.00470).
URL <https://arxiv.org/abs/1910.00470>
- [54] M. Tan, K. Yamaguchi, A. Raney, V. Nockles, M. Leblanc, S. Bendelac, An ai blue team playbook, Assurance and Security for AI-enabled Systems (2024).
URL <https://doi.org/10.1117/12.3021908>
- [55] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, A. Kurakin, On evaluating adversarial robustness (2019). [arXiv:1902.06705](https://arxiv.org/abs/1902.06705).
URL <https://arxiv.org/abs/1902.06705>

Appendix A. Proofs

The proof of Proposition 2.2. Because $V = \mathbb{R}^d$, it is not hard to verify that $\text{cov}(f, f) = \langle f, Af \rangle = \langle A^{1/2}f, A^{1/2}f \rangle = |A^{1/2}f|^2$, where the inner product $\langle \cdot, \cdot \rangle$ is defined on the Euclidean space \mathbb{R}^d . Then we can rewrite Eqn. (1) as follows:

$$\begin{aligned}\|x\|_{\mathcal{L}}^2 &= \sup_{\text{cov}(f,f) \leq 1} f(x)^2 = \sup_{f \in \mathbb{R}^d / \{\mathbf{0}\}} \frac{\langle f, x \rangle^2}{\text{cov}(f, f)} = \sup_{f \in \mathbb{R}^d / \{\mathbf{0}\}} \frac{\langle A^{1/2}f, A^{-1/2}x \rangle^2}{\langle A^{1/2}f, A^{1/2}f \rangle} \\ &\leq \sup_{f \in \mathbb{R}^d / \{\mathbf{0}\}} \frac{\langle A^{1/2}f, A^{1/2}f \rangle \langle A^{-1/2}x, A^{-1/2}x \rangle}{\langle A^{1/2}f, A^{1/2}f \rangle} = \langle A^{-1/2}x, A^{-1/2}x \rangle = \langle x, A^{-1}x \rangle.\end{aligned}$$

On the other hand, for simplicity, assume that the first element of x is nonzero, i.e., $x^1 \neq 0$. Then we can take $f = (|A^{-1/2}x|/x^1, 0, \dots, 0)^T$ and get $f(x) = |A^{-1/2}x|$. Therefore $f(x)^2 \leq \|x\|_{\mathcal{L}}^2$. The assertion is confirmed. \square

Proposition 2.4. By the multivariate Chebyshev inequality, it is easy to deduce that

$$\mathbb{P} \left(|A^{-\frac{1}{2}}(y - \mu)| > \delta \mid y \sim \mathcal{L}_c \right) \leq \frac{d}{\delta^2}.$$

Assume that $\mu_u \neq \mu$. Take $\delta \in (\sqrt{d}, |A^{-\frac{1}{2}}(\mu_u - \mu)|)$, then it suffices to show the inequality (3). Indeed:

$$\begin{aligned}\mathbb{P} \left(|A^{-\frac{1}{2}}(y - \mu)| < \delta \mid y \sim \mathcal{L}_u \right) &= \mathbb{P} \left(|A^{-\frac{1}{2}}(y - \mu_u) + A^{-\frac{1}{2}}(\mu_u - \mu)| < \delta \mid y \sim \mathcal{L}_u \right) \\ &\leq \mathbb{P} \left(|A^{-\frac{1}{2}}(y - \mu_u)| > |A^{-\frac{1}{2}}(\mu_u - \mu)| - \delta \mid y \sim \mathcal{L}_u \right) \leq \frac{d \|A^{-\frac{1}{2}}A_u^{\frac{1}{2}}\|^2}{(|A^{-\frac{1}{2}}(\mu_u - \mu)| - \delta)^2}.\end{aligned}$$

Taking $\delta = \frac{|A^{-\frac{1}{2}}(\mu_u - \mu)|}{1 + \|A^{-\frac{1}{2}}A_u^{\frac{1}{2}}\|}$ reduces to (4). \square

Proposition 2.5. Note that if $x \in \mathcal{L}_c$, then $\mathbb{E}[x + y_i] = 0$ and $\text{Var}(x - y_i) = \text{Var}(X) + \text{Var}(y_i) = 2A$. Therefore,

$$\begin{aligned}\mathbb{P}(\text{dist}(x; \mathcal{L}_c) > \delta \mid x \sim \mathcal{L}_c) &= \mathbb{P} \left(\min_{i \in [n]} \|x - y_i\|_{\mathcal{L}_c} > \delta \mid x \sim \mathcal{L}_c \right) \\ &= \prod_{i \in [n]} \mathbb{P}(\|x - y_i\|_{\mathcal{L}_c} > \delta \mid x \sim \mathcal{L}_c) \leq \prod_{i \in [n]} \frac{2d}{\delta^2} = \left(\frac{2d}{\delta^2} \right)^n.\end{aligned}$$

We shall choose $\delta > \sqrt{2d}$ so the right hand side is smaller than one.

If x has mean μ_u and covariance A_u , then

$$\begin{aligned}\text{Var} \left(A^{-\frac{1}{2}}((\mu_u - x) - (\mu - y_i)) \right) &= \text{Var} \left(A^{-\frac{1}{2}}(\mu_u - x) \right) + \text{Var} \left(A^{-\frac{1}{2}}(\mu - y_i) \right) \\ &= A^{-\frac{1}{2}}A_uA^{-\frac{1}{2}} + A^{-\frac{1}{2}}AA^{-\frac{1}{2}} = A^{-\frac{1}{2}}A_uA^{-\frac{1}{2}} + I.\end{aligned}$$

Now taking $\Delta_\mu := |A^{-\frac{1}{2}}(\mu_u - \mu)|$ and $\Delta_A := A^{-\frac{1}{2}}A_uA^{-\frac{1}{2}} + I$ yields

$$\begin{aligned}\mathbb{P} \left(\|x - y_i\|_{\mathcal{L}_c} < \delta \mid x \sim \mathcal{L}_c \right) &\leq \mathbb{P} \left(|A^{-\frac{1}{2}}((\mu_u - x) - (\mu - y_i))| > \Delta_\mu - \delta \mid x \sim \mathcal{L}_c \right) \leq \frac{d \|\Delta_A^{\frac{1}{2}}\|^2}{(\Delta_\mu - \delta)^2},\end{aligned}$$

where we shall choose $\delta < \Delta_\mu$ and $\frac{d \|\Delta_A^{\frac{1}{2}}\|^2}{(\Delta_\mu - \delta)^2} < 1$. The estimate above can be used to deduce the following:

$$\begin{aligned}\mathbb{P}(\text{dist}(x; \mathcal{L}_c) < \delta \mid x \sim \mathcal{L}_c) &= 1 - \mathbb{P} \left(\min_{i \in [n]} \|x - y_i\|_{\mathcal{L}_c} \geq \delta \mid x \sim \mathcal{L}_c \right) \\ &= 1 - \prod_{i \in [n]} \mathbb{P}(\|x - y_i\|_{\mathcal{L}_c} \geq \delta \mid x \sim \mathcal{L}_c) = 1 - \prod_{i \in [n]} \left(1 - \mathbb{P}(\|x - y_i\|_{\mathcal{L}_c} < \delta \mid x \sim \mathcal{L}_c) \right) \\ &\leq 1 - \left(1 - \frac{d \|\Delta_A^{\frac{1}{2}}\|^2}{(\Delta_\mu - \delta)^2} \right)^n.\end{aligned}$$

\square

Table C.15: Overview of Model Architectures. From left to right: ResNet-18, ResNet-20, RepVGG-A2, and VGG-19. Layers highlighted in red indicate those used for representation extraction in this study.

Layer Type	Output Shape	Layer Type	Output Shape	Layer Type	Output Shape	Layer Type	Output Shape
Input	(3, 32, 32)	Input	(3, 32, 32)	Input	(3, 32, 32)	Input	(3, 32, 32)
Residual Block 1	(64, 16, 16)	Residual Block 1	(16, 32, 32)	RepVGG Block 1	(96, 32, 32)	VGG Block 1	(64, 32, 32)
Residual Block 2	(128, 8, 8)	Residual Block 2	(32, 16, 16)	RepVGG Block 2	(192, 16, 16)	VGG Block 2	(128, 16, 16)
Residual Block 3	(256, 4, 4)	Residual Block 3	(64, 8, 8)	RepVGG Block 3	(384, 8, 8)	VGG Block 3	(256, 8, 8)
Residual Block 4	(512, 2, 2)	Pooling Layer	(64, 1, 1)	RepVGG Block 4	(1408, 4, 4)	VGG Block 4	(512, 4, 4)
Pooling Layer	(512, 1, 1)	Fully Connected	(10)	Pooling Layer	(1408, 1, 1)	VGG Block 5	(512, 2, 2)
Fully Connected	(10)			Fully Connected	(10)	Pooling layer	(512, 1, 1)
						Fully Connected	(10)

ResNet-18

ResNet-20

RepVGG-A2

VGG-19

Appendix B. The discretization for the level 2 of 2D-signature

The 2nd level of S contains $4d^2$ elements

$$\{\mathrm{d}x^i \mathrm{d}x^j, \mathrm{d}x^i \hat{\mathrm{d}}x^j, \hat{\mathrm{d}}x^i \mathrm{d}x^j, \hat{\mathrm{d}}x^i \hat{\mathrm{d}}x^j, i, j \in [d]\}$$

of four types, where the first two reinforce the effects of the same type and the last two capture interaction between different effects:

$$\begin{aligned} & \int \int_{(0,0)}^{(1,1)} \left(\int \int_{(0,0)}^{(s_1, t_1)} \frac{\partial^2 x^i(s_2, t_2)}{\partial s_2 \partial t_2} \mathrm{d}s_2 \mathrm{d}t_2 \right) \frac{\partial^2 x^i(s_1, t_1)}{\partial s_1 \partial t_1} \mathrm{d}s_1 \mathrm{d}t_1 \\ & \approx \sum_{k_1=1}^{N-1} \sum_{k_2=1}^{M-1} \left[\left(\sum_{k_3=1}^{k_1-1} \sum_{k_4=1}^{k_2-1} (\mathbf{x}_{k_3+1, k_4+1}^i - \mathbf{x}_{k_3, k_4+1}^i - \mathbf{x}_{k_3+1, k_4}^i + \mathbf{x}_{k_3, k_4}^i) \right) \right. \\ & \quad \times \left. (\mathbf{x}_{k_1+1, k_2+1}^j - \mathbf{x}_{k_1, k_2+1}^j - \mathbf{x}_{k_1+1, k_2}^j + \mathbf{x}_{k_1, k_2}^j) \right] \\ & = \sum_{k_1=1}^{N-1} \sum_{k_2=1}^{M-1} \left[\left(\mathbf{x}_{k_1, k_2}^i - \mathbf{x}_{k_1, 1}^i - \mathbf{x}_{1, k_2}^i + \mathbf{x}_{1, 1}^i \right) \right. \\ & \quad \times \left. (\mathbf{x}_{k_1+1, k_2+1}^j - \mathbf{x}_{k_1, k_2+1}^j - \mathbf{x}_{k_1+1, k_2}^j + \mathbf{x}_{k_1, k_2}^j) \right]; \end{aligned}$$

and

$$\begin{aligned} & \int \int_{(0,0)}^{(1,1)} \left(\int \int_{(0,0)}^{(s_1, t_1)} \frac{\partial x^i(s_2, t_2)}{\partial s_2} \frac{\partial x^i(s_2, t_2)}{\partial t_2} \mathrm{d}s_2 \mathrm{d}t_2 \right) \frac{\partial x^i(s_1, t_1)}{\partial s_1} \frac{\partial x^i(s_1, t_1)}{\partial t_1} \mathrm{d}s_1 \mathrm{d}t_1 \\ & \approx \sum_{k_1=1}^{N-1} \sum_{k_2=1}^{M-1} \left[\left(\sum_{k_3=1}^{k_1-1} \sum_{k_4=1}^{k_2-1} (\mathbf{x}_{k_3+1, k_4}^i - \mathbf{x}_{k_3, k_4}^i) \times (\mathbf{x}_{k_3, k_4+1}^i - \mathbf{x}_{k_3, k_4}^i) \right) \right. \\ & \quad \times \left. (\mathbf{x}_{k_1+1, k_2}^j - \mathbf{x}_{k_1, k_2}^j) \times (\mathbf{x}_{k_1, k_2+1}^j - \mathbf{x}_{k_1, k_2}^j) \right]. \end{aligned}$$

Similarly, for the latter two we shall have the following approximation

$$\begin{aligned} & \sum_{k_1=1}^{N-1} \sum_{k_2=1}^{M-1} \left[\left(\sum_{k_3=1}^{k_1-1} \sum_{k_4=1}^{k_2-1} (\mathbf{x}_{k_3+1, k_4+1}^i - \mathbf{x}_{k_3, k_4+1}^i - \mathbf{x}_{k_3+1, k_4}^i + \mathbf{x}_{k_3, k_4}^i) \right) \right. \\ & \quad \times \left. (\mathbf{x}_{k_1+1, k_2+1}^j - \mathbf{x}_{k_1, k_2+1}^j - \mathbf{x}_{k_1+1, k_2}^j + \mathbf{x}_{k_1, k_2}^j) \right]; \end{aligned}$$

and

$$\begin{aligned} & \sum_{k_1=1}^{N-1} \sum_{k_2=1}^{M-1} \left[\left(\sum_{k_3=1}^{k_1-1} \sum_{k_4=1}^{k_2-1} (\mathbf{x}_{k_3+1, k_4+1}^i - \mathbf{x}_{k_3, k_4+1}^i - \mathbf{x}_{k_3+1, k_4}^i + \mathbf{x}_{k_3, k_4}^i) \right) \right. \\ & \quad \times \left. (\mathbf{x}_{k_1+1, k_2}^j - \mathbf{x}_{k_1, k_2}^j) (\mathbf{x}_{k_1, k_2+1}^j - \mathbf{x}_{k_1, k_2}^j) \right] \\ & = \sum_{k_1=1}^{N-1} \sum_{k_2=1}^{M-1} \left[\left(\mathbf{x}_{k_1, k_2}^i - \mathbf{x}_{k_1, 1}^i - \mathbf{x}_{1, k_2}^i + \mathbf{x}_{1, 1}^i \right) \right. \\ & \quad \times \left. (\mathbf{x}_{k_1+1, k_2}^j - \mathbf{x}_{k_1, k_2}^j) (\mathbf{x}_{k_1, k_2+1}^j - \mathbf{x}_{k_1, k_2}^j) \right]. \end{aligned}$$

Appendix C. Tables and plots

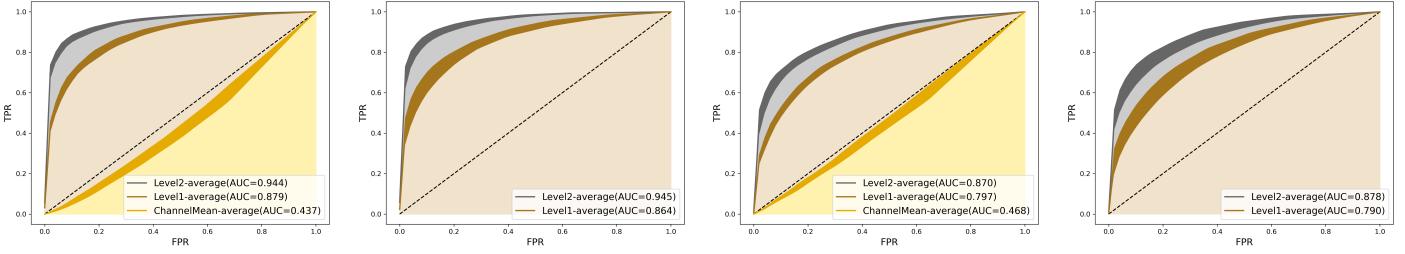


Figure C.17: AUROCs of detecting untargeted IFGSM attack from RepVGG-A2 with noise levels $\varepsilon = 0.03$ and $\varepsilon = 0.02$. From left to right: 2DSig-Norm ($\varepsilon = 0.03$), 2DSig-Conf ($\varepsilon = 0.03$), 2DSig-Norm ($\varepsilon = 0.02$), 2DSig-Conf ($\varepsilon = 0.02$).

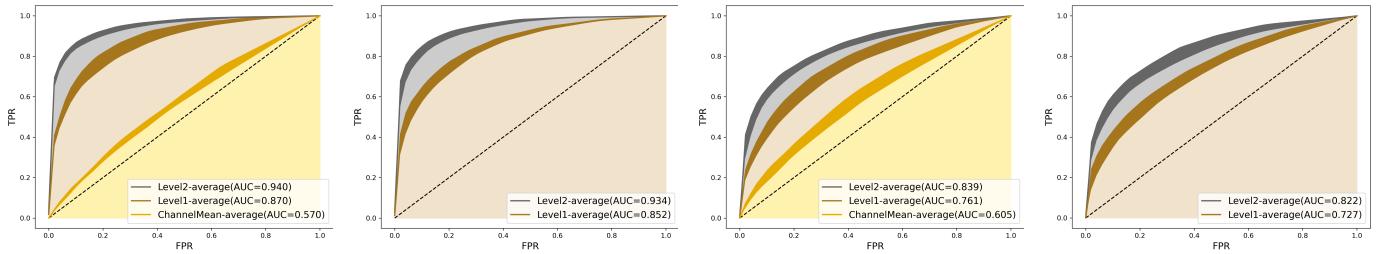


Figure C.18: AUROCs of detecting targeted IFGSM attack from RepVGG-A2 with noise levels $\varepsilon = 0.03$ and $\varepsilon = 0.02$. From left to right: 2DSig-Norm ($\varepsilon = 0.03$), 2DSig-Conf ($\varepsilon = 0.03$), 2DSig-Norm ($\varepsilon = 0.02$), 2DSig-Conf ($\varepsilon = 0.02$).

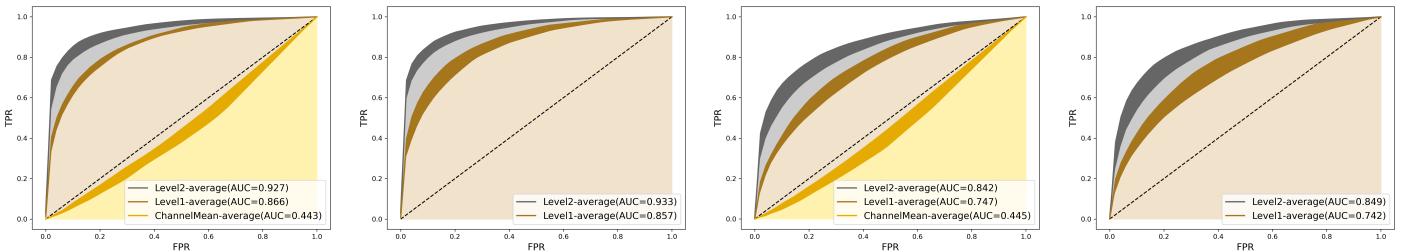


Figure C.19: AUROCs of detecting untargeted IFGSM attack from RepVGG-A2 with noise levels $\varepsilon = 0.03$ and $\varepsilon = 0.02$. From left to right: 2DSig-Norm ($\varepsilon = 0.03$), 2DSig-Conf ($\varepsilon = 0.03$), 2DSig-Norm ($\varepsilon = 0.02$), 2DSig-Conf ($\varepsilon = 0.02$).

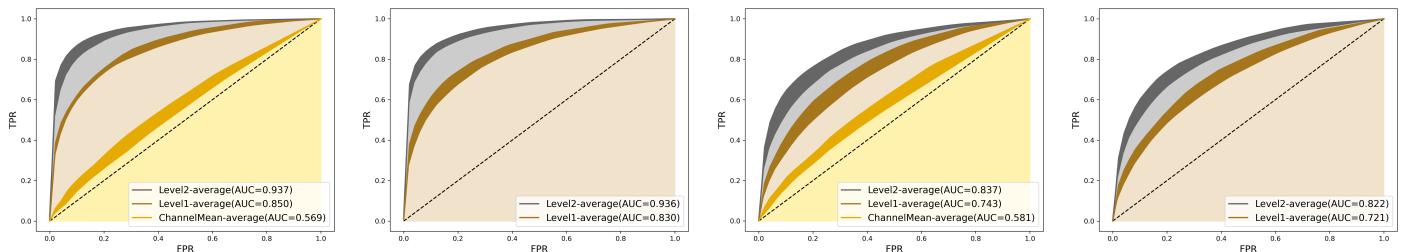


Figure C.20: AUROCs of detecting targeted IFGSM attack from ResNet-20 with noise levels $\varepsilon = 0.03$ and $\varepsilon = 0.02$. From left to right: 2DSig-Norm ($\varepsilon = 0.03$), 2DSig-Conf ($\varepsilon = 0.03$), 2DSig-Norm ($\varepsilon = 0.02$), 2DSig-Conf ($\varepsilon = 0.02$).

Table C.16: The F1-score and TPR for the 2DSig-Norm to defend against PGDM attacks with different noise levels ε .

	RepVGG-A2		ResNet-20		
	Untarget	Target	Untarget	Target	
F1-score	$\varepsilon = 0.03$	0.88 ± 0.01	0.89 ± 0.01	0.88 ± 0.01	0.88 ± 0.02
	$\varepsilon = 0.02$	0.86 ± 0.01	0.86 ± 0.01	0.84 ± 0.01	0.86 ± 0.01
TPR	$\varepsilon = 0.03$	0.99 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	1.00 ± 0.00
	$\varepsilon = 0.02$	0.94 ± 0.01	0.95 ± 0.01	0.92 ± 0.01	0.96 ± 0.01

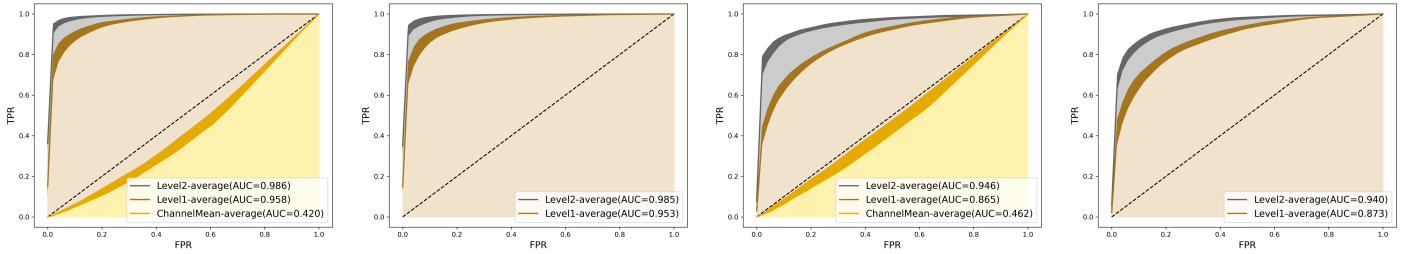


Figure C.21: AUROCs of detecting untargeted PGDM attack from RepVGG-A2 with noise levels $\varepsilon = 0.03$ and $\varepsilon = 0.02$. From left to right: 2DSig-Norm ($\varepsilon = 0.03$), 2DSig-Conf ($\varepsilon = 0.03$), 2DSig-Norm ($\varepsilon = 0.02$), 2DSig-Conf ($\varepsilon = 0.02$).

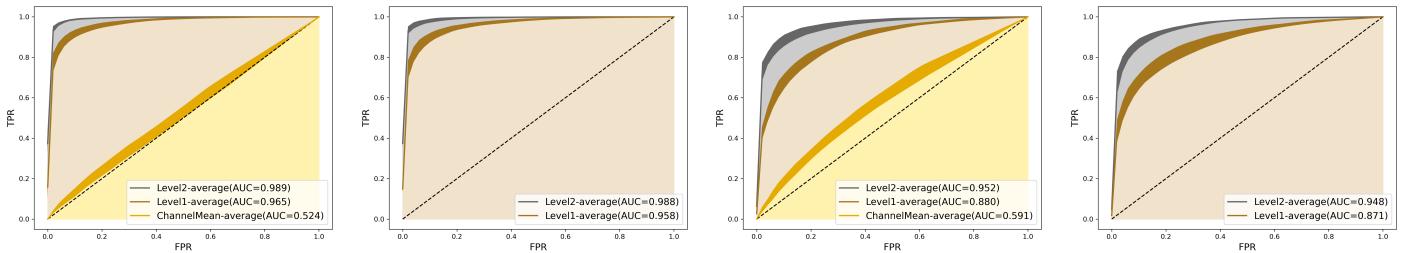


Figure C.22: AUROCs of detecting targeted PGDM attack from RepVGG-A2 with noise levels $\varepsilon = 0.03$ and $\varepsilon = 0.02$. From left to right: 2DSig-Norm ($\varepsilon = 0.03$), 2DSig-Conf ($\varepsilon = 0.03$), 2DSig-Norm ($\varepsilon = 0.02$), 2DSig-Conf ($\varepsilon = 0.02$).

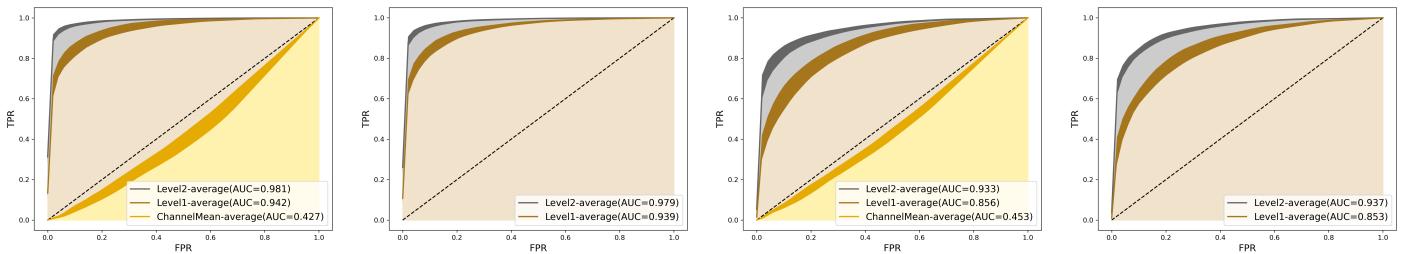


Figure C.23: AUROCs of detecting untargeted PGDM attack from ResNet-20 with noise levels $\varepsilon = 0.03$ and $\varepsilon = 0.02$. From left to right: 2DSig-Norm ($\varepsilon = 0.03$), 2DSig-Conf ($\varepsilon = 0.03$), 2DSig-Norm ($\varepsilon = 0.02$), 2DSig-Conf ($\varepsilon = 0.02$).

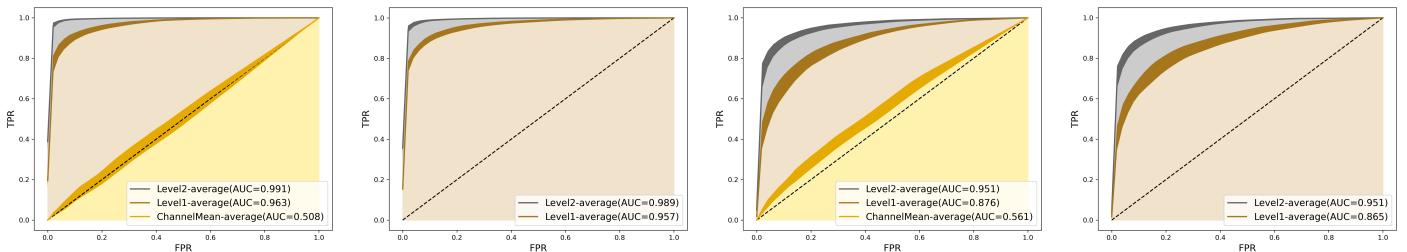


Figure C.24: AUROCs of detecting targeted PGDM attack from ResNet-20 with noise levels $\varepsilon = 0.03$ and $\varepsilon = 0.02$. From left to right: 2DSig-Norm ($\varepsilon = 0.03$), 2DSig-Conf ($\varepsilon = 0.03$), 2DSig-Norm ($\varepsilon = 0.02$), 2DSig-Conf ($\varepsilon = 0.02$).