Applied Intelligence (2023) 53:509–531
https://doi.org/10.1007/s10489-022-03373-y

Revisiting model's uncertainty and confidences for adversarial example detection

Ahmed Aldahdooh[1] · **Wassim Hamidouche**[1] · **Olivier Déforges**[1]

Accepted: 9 February 2022 / Published online: 19 April 2022

Abstract

Security-sensitive applications that rely on Deep Neural Networks (DNNs) are vulnerable to small perturbations that are crafted to generate Adversarial Examples. The (AEs) are imperceptible to humans and cause DNN to misclassify them. Many defense and detection techniques have been proposed. Model's confidences and Dropout, as a popular way to estimate the model's uncertainty, have been used for AE detection but they showed limited success against black- and gray-box attacks. Moreover, the state-of-the-art detection techniques have been designed for specific attacks or broken by others, need knowledge about the attacks, are not consistent, increase model parameters overhead, are time-consuming, or have latency in inference time. To trade off these factors, we revisit the model's uncertainty and confidences and propose a novel unsupervised ensemble AE detection mechanism that 1) uses the uncertainty method called SelectiveNet, 2) processes model layers outputs, i.e. feature maps, to generate new confidence probabilities. The detection method is called SFAD. Experimental results show that the proposed approach achieves better performance against black- and gray-box attacks than the state-of-the-art methods and achieves comparable performance against white-box attacks. Moreover, results show that SFAD is fully robust against High Confidence Attacks (HCAs) for MNIST and partially robust for CIFAR10 datasets.[1]

Keywords Adversarial examples · Adversarial attacks · Adversarial example detection · Deep learning robustness

1 Introduction

DL has achieved remarkable advances in different fields in human life especially computer vision tasks like object detection, image classification [1–3], surveillance [4], and medical imaging [5]. Despite that, it is found that DL models are vulnerable to adversaries [6, 7]. In image classification models, for instance, adversaries can generate AEs, by adding small perturbations to an input image that are imperceptible to humans and devices, that cause DL models to misclassify the input images. Such potential threat affects security-critical DL-based applications [8] such as self-driving cars.

Adversaries can generate AEs for white-box, black-box, and gray-box attacks [9, 10]. In white-box attack scenario, the adversary knows everything about the DL-model including inputs, outputs, architecture, and weights of the model. Hence, he is guided by the model gradient to generate AE by solving an optimization problem [7, 11–14]. In black-box scenario, the adversary knows nothing about the model but he leverages the transferability property [15] of AEs and the input content. By sending queries to the model, the adversary can craft small perturbations that are harmonious with the input image [16–19]. In the gray-box scenario, the adversary knows only the input and the output of the model and hence, he tries to substitute the original model with an approximated model and then uses its gradient as in white-box scenario to generate AEs.

Researchers pay attention to this threat and several emerging methods have been proposed to detect or to defend against AEs. More details about defense and detection methods can be found in Section 2.

The source code is available in https://aldahdooh.github.io/SFAD/.

✉ Ahmed Aldahdooh
ahmed.aldahdooh@insa-rennes.fr

Wassim Hamidouche
wassim.hamidouche@insa-rennes.fr

Olivier Déforges
olivier.deforges@insa-rennes.fr

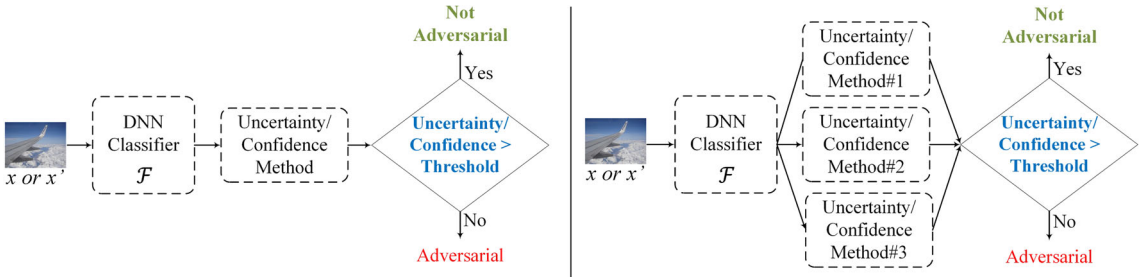
[1] INSA Rennes, CNRS, University of Rennes, IETR - UMR 6164, F-35000 Rennes, France

DL model's uncertainty is one of the main methods that has been used to determine whether an input sample belongs to the training manifold. The uncertainty is usually measured by adding randomness to the model using Dropout technique [20, 21]. It is found that clean sample predictions do not change, when randomness is added, while it changes for AEs. Feinman et al. [22] proposed BU metric that used Monte Carlo dropout to estimate the uncertainty to detect AEs that are near the classes manifold, while Smith et al. [23, 24] used mutual information method to estimate the uncertainty. The prediction risk of these methods is higher compared to the recent uncertainty method, SelectiveNet [25], that is used in this work. On the other hand, it was shown in [26] that predicted class probabilities, i.e. model's confidence, of in-of-distribution samples are higher than of out-of-distribution. Model's confidence was used in [26–29] to implement AE detectors. Uncertainty and confidence based detectors showed limited success against black- and gray-box attacks. Uncertainty and confidence based detectors are usually threshold-based detectors as shown in Fig. 1(a). To enhance detectors' performance, one recommendation goes to the direction of providing ensemble detection methods, as shown in Fig. 1b. Although state-of-the-art detectors achieve promising results, they may have one or more limitation(s); not performing well with some known attacks [30], broken by attackers [31, 32], performance of baseline detectors is not consistent [33], increase the model parameters overhead [34], time consuming [35], or introduce latency [36] in the inference time.

In this paper and in order to mitigate the aforementioned limitations, we revisit the model's uncertainty and confidence to propose a novel ensemble AE detector that hasn't had any knowledge of AEs, i.e. unsupervised detector, as shown in Fig. 2. The proposed method has the following attributes; 1) it investigates SelectiveNet capability in detecting adversarial examples since it measures the uncertainty with less risk. According to the author's knowledge, the SlelectiveNet [25] is not used in adversarial attacks detection models. 2) Unlike other detectors [29, 37, 38], the proposed method uses the model's last N-layers outputs, i.e. feature maps, to build N-CNNs $\mathcal{M}$ that have different processing blocks like up/down sampling, auto-encoders [39, 40], noise addition [41–43], and bottleneck layer addition [44] that make the representative data of last layers more unique to the input data distribution to yield better model's confidence. To reduce the effect of white-box attacks, the output of $\mathcal{M}$ is transferred/distilled to build the last CNN $\mathcal{S}$. 3) The proposed model ensembles the proposed detection techniques to provide the final detector. This step has a great impact in reducing the adversary's capability to craft perturbations that can fool the detector, since he has to fool every detection technique. We name the proposed method as Selective and Feature based Adversarial Detection (SFAD). The high-level architecture of the SFAD is illustrated in Fig. 1b.

A prototype of SFAD is tested under white-box, black-box and gray-box attacks on MNIST [45], and CIFAR10 [46]. Under the white-box attacks, the experimental results show that SFAD can detect AEs at least with accuracy of 89.8% (many with 99%) for all tested attacks except for the PGD attack [14] with at least 65% detection accuracy in average. For black- and gray-box attacks, SFAD shows better performance than other tested detectors. Finally, SFAD is tested under the HCA [47] and it shows that it is fully(100%) and partially(57.76%) robust on MNIST and CIFAR10 respectively. SFAD sets the thresholds to reject 10% of clean images. Moreover, comparisons with state-of-the-art methods are presented. Hence, our key contributions are:

– We propose a novel unsupervised ensemble model for AE detection. Ensemble detection makes SFAD more robust against white-box and adaptive attacks.



(a) Uncertainty/Confidence-based detection

(b) Uncertainty/Confidence-based Ensemble detection

Fig. 1 **a** High-level architecture of the uncertainty/confidence-based detectors. **b** High-level architecture of the uncertainty/confidence-based ensemble detectors. The input sample is passed to the CNN model to do class prediction. The detector, i.e. the uncertainty method, estimates the uncertainty of the input samples using model hidden layers. Using a predefined threshold, the input sample is not adversarial if the uncertainty exceeds the predefined threshold

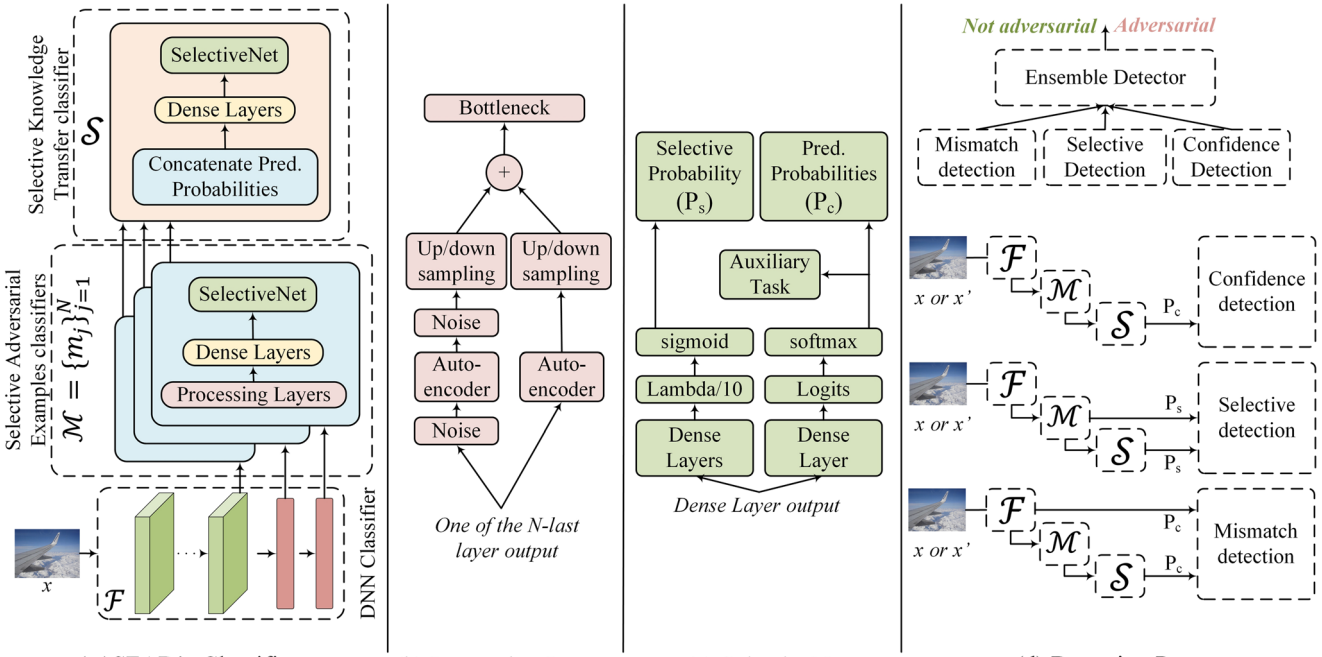


Fig. 2 **a** SFAD's architecture. N-last representative output of DNN is used to build N Selective Adversarial Example classifiers. The confidence output, i.e. pred. probabilities, of the N classifiers is concatenated to be as input for Selective Knowledge Transfer classifier. **b** Feature maps processing blocks. **c** SelectiveNet architecture [25] **d** Detection Process: Selective probabilities ($P_s^{m_j}$ and $P_s^{\mathcal{S}}$) are used in the Selective detection process. Confidence/Prediction probabilities, ($P_c^{m_j}$ and $P_c^{\mathcal{S}}$), are used in the confidence detection process. Confidence/Prediction probabilities, ($P_c^{\mathcal{F}}$ and $P_c^{\mathcal{S}}$), are used in the mismatch detection process. The total detection is the ensemble of the three detection modules

- We investigate the SelectiveNet's, as an uncertainty model, capability in detecting AEs.
- We show that, by processing the feature maps of last N-layers, we can build classifiers for better confidence distribution. We provide an ablation experiments to study the impact of the feature processing blocks.
- SFAD prototype proves the concept of the approach and lets the door open in future to find the best N layers and the best N(or M) CNNs combinations to build the detector's classifiers.
- Unlike tested state-of-the-art detectors, SFAD prototype shows better performance under gray- and black-box attacks. SFAD prototype shows that it is fully robust on MNIST and partially robust on CIFAR10 when attacked with HCAs. For instance, Local Intrinsic Dimensionality (LID) method [48] reported very high detection accuracy on the tested attacks, but fails on HCAs [31, 47].

2 Related work

2.1 Detection methods

Defense techniques like adversarial training [7, 14, 49, 50], feature denoising [51–53], pre-processing [54, 55], and gradient masking [56–59] try to make the model robust against the attacks and let the model correctly classify the AEs. On the other hand, detection methods provide adversarial status for the input image. Detection techniques can be classified according to the presence of AEs in the detector learning process into supervised and unsupervised techniques [33]. In supervised detection, detectors include AEs in the learning process. Many approaches exist in the literature. In the feature-based approach [38, 60–63], detectors use clean and AEs inputs to built their classifier models from scratch by using raw image data or by using the representative layers' outputs of a DNN model. For instance, in [38], the detector quantizes the last ReLU activation layer of the model and builds a binary v classifier. As reported in [38], this detector is not robust enough and is not tested against strong attacks like Carlini-Wagner (CW) attacks. While the work in [61] added a new adversarial class to the NN model and train the model from scratch with clean and adversarial inputs. This architecture reduces the model accuracy [61]. In the concurrent recent work [63], Wang et al. used the saliency map features of clean and adversarial examples to learn the classifier's detector. In the statistical-based approach [22, 48], detectors perform statistical measurement to define the separation between clean and adversarial inputs. In [22], KD estimation, BU, or

combined models are introduced. Kernel-density feature is extracted from clean and AEs in order to identify AEs that are far away from data manifold while Bayesian uncertainty feature identifies the AEs that lie in low-confidence regions of the input space. LID method is introduced in [48] as a distance distribution of the input sample to its neighbors to assess the space-filling capability of the region surrounding that input sample. The works in [31, 47] showed that these methods can be broken. Finally, the network invariant approach [62, 64] learns the differences in neuron activation values between clean input samples and AEs to build a binary NN detector. The main limitation of this approach is that it requires prior knowledge about the attacks and hence it might not be robust against new or unknown attacks.

On the other hand, in unsupervised detection, detectors are trained with clean images only to identify the AEs. It is also known as prediction inconsistency models since it depends on the fact that AEs might not fool every NN model. That's because the input feature space is almost limited and the adversary always takes that as an advantage to generate the AEs. Hence, unsupervised detectors try to reduce this limited input feature space available to adversaries. Many approaches have been presented in the literature. The Feature Squeezing (FS) approach [30] measures the distance between the predictions of the input and the same input after squeezing. The input will be adversarial if the distance exceeds a threshold. The work in [30] squeezes out unnecessary input features by reducing the color bit depth of each pixel and by spatial smoothing of adversarial inputs. As reported in [30], FS is not performing well with some known attacks like FGSM. Instead of squeezing, denoising based approach, like MagNet [65], measures the distances between the predictions of input samples and denoised/filtered input samples. It was found in [32, 53] that MagNet can be broken and do not scale to large images. Recently, a network invariant approach was introduced [35]. They proposed a NIC method that builds a set of models for individual layers to describe the provenance and the activation value distribution channels. It was observed that AEs affect these channels. The provenance channel describes the instability of activated neurons set in the next layer when small changes are present in the input sample while the activation value distribution channel describes the changes with the activation values of a layer. The reported performance of this method showed its superiority against other state-of-the-art models but other works reported that the baseline NIC's detectors are not consistent [33], increase model parameters overhead [34], are time consuming [35], and increase the latency in the inference time [36].

Uncertainty-based detectors Following the observation that the prediction of clean image remains correct with many dropouts, while the prediction of AE changes. Feinman et al. [22] proposed BU metric. BU uses Monte Carlo dropout to estimate the uncertainty, to detect those AEs that are near the classes manifold, while Smith et al. [23] used mutual information method for such a task. In [24], Sheikholeslami et al. proposed an unsupervised detection method that provides a layer-wise minimum variance solver to estimate model's uncertainty for in-distribution training data. Then, a mutual information based threshold is identified.

Confidence-based detectors Aigrain et al. [27] built a simple NN detector that uses the model's logits of clean and AEs to build a binary classifier. Inspired by the hypothesis of that, for a given perturbed image, different models yield different confidences, Monteiro et al. [28] proposed a bi-model mismatch detection method. The detector is a binary RBF-SVM classifier that takes as input the output of two classifiers of clean and AEs. On the other hand, Sotgiu et al. proposed an unsupervised detection method that uses the last N representative layers' outputs of the classifier to built three SVM classifiers with RBF kernel. The confidence probabilities of the SVMs are combined to build the last SVM-RBF classifier. Then, a threshold is identified to reject inputs that have less maximum confidence probability.

2.2 SelectiveNet as an uncertainty model

Let $\mathcal{X}$ be an input space, e.g. images, and $\mathcal{Y}$ a label space. Let $\mathbb{P}(X, Y)$ be the data distribution over $\mathcal{X} \times \mathcal{Y}$. A model, $f : X \to Y$, is called a prediction function, $\ell : Y \times Y \to \mathbb{R}^2$ is a given loss function. Given a labeled set $S_k = (x_i, y_i)_{i=1}^k \subseteq (\mathcal{X} \times \mathcal{Y})^k$ sampled i.i.d. from $\mathbb{P}(X, Y)$, where k is the number of training samples. The true risk of the prediction function f w.r.t. $\mathbb{P}$ is $R(f) \triangleq \mathbb{E}_{\mathbb{P}(X,Y)}[\ell(f(x), y)]$ while the empirical risk of the prediction function f is $\hat{r}(f \mid S_k) \triangleq \frac{1}{k} \sum_{i=1}^k \ell(f(x_i), y_i)$.

Here, we briefly demonstrate the SelectiveNet as stated in [25]. The selective model is a pair (f, g), where f is a prediction function, and $g : \mathcal{X} \to \{0, 1\}$ is a binary selection function for f,

$$(f, g)(x) \triangleq \begin{cases} f(x), & \text{if } g(x) = 1; \\ \text{don't know}, & \text{if } g(x) = 0. \end{cases} \tag{1}$$

A soft selection function can also be considered, where $g : \mathcal{X} \to [0, 1]$, hence, the value of $(f, g)(x)$ is calculated with the help of a threshold τ as expressed in the following equation

$$(f, g)(x) \triangleq \begin{cases} f(x), & \text{if } g(x) \geq \tau; \\ \text{don't know}, & \text{if } g(x) < \tau. \end{cases} \tag{2}$$

The performance of a selective model is calculated using coverage and risk. The true coverage is defined to be

the probability mass of the non-rejected region in $\mathcal{X}$ and calculated as

$$\phi(g) \triangleq E_P[g(x)], \tag{3}$$

while the empirical coverage is calculated as

$$\hat{\phi}(g \mid S_k) \triangleq \frac{1}{k}\sum_{i=1}^{k} g(x_i) \tag{4}$$

The true selective risk of (f, g) is

$$R(f, g) \triangleq \frac{E_P[\ell(f(x), y)g(x)]}{\phi(g)}, \tag{5}$$

while the empirical selective risk is calculated for any given labeled set S_k as

$$\hat{r}(f, g \mid S_k) \triangleq \frac{\frac{1}{k}\sum_{i=1}^{k} \ell(f(x_i), y_i)g(x_i)}{\hat{\phi}(g \mid S_k)}. \tag{6}$$

Finally, for a given coverage rate $0 < c \leq 1$ and Θ, a set of parameters for a given deep network architecture for f and g, the optimization problem of the selective model is expressed as:

$$\begin{aligned} \theta^* = \arg\min_{\theta\in\Theta}(R(f_\theta, g_\theta)) \\ s.t.\ \phi(g_\theta) \geq c, \end{aligned} \tag{7}$$

and can be solved using the Interior Point Method (IPM) [66] to enforce the coverage constraint. That yields to unconstrained loss objective function over samples in S_k,

$$\begin{aligned} \mathcal{L}_{(f,g)} \triangleq \hat{r}_\ell(f, g \mid S_k) + \lambda\Psi(c - \hat{\phi}(g \mid S_k)) \\ \Psi(a) \triangleq \max(0, a)^2, \end{aligned} \tag{8}$$

where c is the target coverage, λ is a hyper-parameter controlling the relative importance of the constraint, and Ψ is a quadratic penalty function. As a result, SelectiveNet is a selective model (f, g) that optimizes both $f(x)$ and $g(x)$ in a single model in a multi-task setting as depicted in Fig. 2c. For more details about the SelectiveNet model, readers are advised to read [25].

3 Adversarial Detection (SFAD) method

3.1 SFAD's classifiers design

It is believed that the last N layers in the DNN $\mathcal{F}$ have potentials in detecting and rejecting AEs [29, 37]. In [67] and [37], only the last layer ($N = 1$) is utilized to detect AEs. At this very high level of presentation, AEs are indistinguishable from samples of the target class. This observation is enhanced when DNR [29] used the last three layers to build SVM with RBF kernel based classifiers. Unlike other works, in this work, 1) feature maps of the last layers Z_j, where $j = \{1, 2 \ldots, N\}$, are processed. In the aforementioned methods, the representatives of the last layers are not processed and basically the detectors represent another approximation of the baseline classifier which is considered as a weak point. 2) MTL is used via the SelectiveNet. MTL has an advantage of combining related tasks with one or more loss function(s) and it does better generalization especially with the help of the auxiliary functions. For more details about MTL, please refer to these recent review papers [68, 69].

In this section, the Adversarial Detection (SFAD) method is demonstrated. As depicted in Fig. 2a, SFAD consists of two main blocks; the selective AEs classifiers $\mathcal{M}$ block (in blue), where $\mathcal{M} = \{m_j\}_{j=1}^{N}$, and the selective knowledge transfer classifier $\mathcal{S}$ block (in orange). In the training phase, we have two steps; the first is to train the $\mathcal{M}$ classifiers, and the second step is to train the $\mathcal{S}$ classifier. Hence, $\mathcal{M}$, and $\mathcal{S}$ are trained separately. While in the inference/test time, the output of $\mathcal{F}$, $\mathcal{M}$, and $\mathcal{S}$ blocks, i.e. model's uncertainties and confidences, are used in the detection process, as depicted in Fig. 2d.

3.2 Selective AEs classifiers block: training the $\mathcal{M}$ classifiers

As shown in Fig. 2a, the aim of $\mathcal{M}$ block is to build N individual classifiers, $\mathcal{M} = \{m_j\}_{j=1}^{N}$. It was shown that perturbation propagation becomes clear when the DNN model goes deeper, hence, using N-last layers have potential in identifying the AEs. Unlike works in [29, 37], we process the representative last N-layer(s) outputs Z_j in different ways in order to make clean input features more unique, as shown in Fig. 2b and discussed in the next Section 3.2.1. This will limit the feature space that the adversary uses to craft the AEs [30, 65]. Moreover, each of the last N-layer output has its own feature space which makes each m_j classifier be trained with different feature space. Hence, combining and increasing the number of N will enhance the detection process.

For simplicity and as recommended in [29], we set $N = 3$ in the implemented prototype and hence, each individual layer output is assigned to a classifier as shown in Fig. 2a. Let the last N layers' outputs z_{ji} of x_i from S_k are z_{1i}, z_{2i}, and z_{Ni}, respectively, where, $j = \{1, 2, \ldots, N\}$. z_{ji} are individually the inputs of the m_j classifier.

The outputs of the m_j classifier are denoted as $m_{ji}(z_{ji})$. Let $\mathcal{Y}' = \mathcal{Y} + 1$ be a label space of m_j, where the extra label is denoted for the selective status, hence m_j represents a function $m_j : Z_j \rightarrow Y'$ on a distribution $\mathbb{P}(Z_j, Y')$ over $\mathcal{Z} \times \mathcal{Y}'$. We refer to the selective probability of m_j as $P_s^{m_j}$ and the confidence probabilities of m_j as $P_c^{m_j}$. m_j optimizes the overall loss function

$$\mathcal{L}_{m_j} = \alpha\mathcal{L}_{(m_j, g_{m_j})} + (1 - \alpha)\mathcal{L}_{h_{m_j}}, \text{ where } \alpha = 0.5, \tag{9}$$

where $\mathcal{L}_{(m_j,g_{m_j})}$ is the selective loss function of m_j, as discussed in Section 2.2, and $\mathcal{L}_{h_{m_j}}$ is the auxiliary loss function of m_j and are calculated as following:

$$\mathcal{L}_{(m_j,g_{m_j})} \triangleq \hat{r}_\ell(m_j, g_{m_j} \mid S_k) + \lambda\, \Psi(c - \hat{\phi}(g_{m_j} \mid S_k)),$$
$$\Psi(a) \triangleq \max(0, a)^2,$$

$$\mathcal{L}_{h_{m_j}} = \hat{r}(h_{m_j} \mid S_k) = \frac{1}{k}\sum_{i=1}^{k} \ell(h_m(z_{ji}), y_i).$$

Studying the value of α is out of the paper scope, but other task balancing methods, may be applied like, uncertainty [70], GradNorm [71], DWA [40], DTP [72], and MGDA [73].

3.2.1 Feature maps processing

As depicted in Fig. 2b each selective classifier consists of different processing blocks; auto-encoder block, up/down-sampling block, bottleneck block, and noise block. These blocks aim at giving distinguishable features for input samples to let the detector recognize the AEs efficiently.

Auto-encoder Auto-encoders are widely used as a reconstruction tool and its loss is used as a score for different tasks. For instance, it is used in the detection process of AEs in [65]. It is believed that AEs gave higher reconstruction loss than clear images. This process is a.k.a attention mechanism [74, 75] and it is used to focus on better representation of input features especially on the shallow classifiers.

Up/down-sampling Up sampling and down sampling are used in different deep classifiers [39, 40]. The aim of down sampling, a.k.a pooling layers in NN, is to gather the global information of the input signal. Hence, if we consider the clean input signal as a signal that has global information and then we expand the global information by bi-linear up sampling and then down sample by average pooling, we will measure the ability of global information reconstruction of the input signal. Besides, this process can be seen as a use case of the reconstruction process.

Noise Adding noise has a potential impact in making NN more robust against AEs and it has been used in many defense methods [41–43]. In this work, we add a branch in the classifier that adds small Gaussian noise to the input signal before and after the auto-encoder block. Then, the noised and clean input features are concatenated before the bottleneck block.

Bottleneck The bottleneck block [44] consists of three convolutional layers; 1×1, 3×3, and 1×1 convolutional layers. The bottleneck name came from the fact that the 3×3 convolutional layer is left as a bottleneck between 1×1 convolutional layers. It is mainly designed for efficiency purposes but according to [74, 76] it is very effective in building shallow classifiers which helps having better representation of input signal.

3.3 Selective knowledge transfer block: training the $\mathcal{S}$ classifier

The block $\mathcal{S}$ aims at building selective knowledge transfer classifier. It concatenates the confidence values of Y classes of the $\mathcal{M}$ classifiers. The idea behind the block $\mathcal{S}$ is that each set of its input is considered as a special feature of the clean input. Hence, we transfer this knowledge, m_j confidence probabilities, of clean inputs to the classifier. Besides, in the inference time, we believe that AE will generate a different distribution of the confidence values and if the AE is able to fool one m_j, it may not fool the others.

As Fig. 2a shows, the confidence probabilities of m_j classifiers are concatenated to be as an input $Q = concat(P_c^{m_1}, P_c^{m_2}, \ldots, P_c^{m_N})$ for the selective knowledge transfer block $\mathcal{S}$. The $\mathcal{S}$ classifier consists of one or more dense layer(s) and yields the selective probability of $\mathcal{S}$ as $P_s^{\mathcal{S}}$ and the confidence probabilities of $\mathcal{S}$ as $P_c^{\mathcal{S}}$. $\mathcal{S}$ represents a function $\mathcal{S} : Q \rightarrow Y'$ on a distribution $\mathbb{P}(Q, Y')$ over $\mathcal{Q} \times \mathcal{Y}'$. Hence, it optimizes the following loss function

$$\mathcal{L}_{\mathcal{S}} = \alpha \mathcal{L}_{(\mathcal{S},g_{\mathcal{S}})} + (1-\alpha)\mathcal{L}_{h_{\mathcal{S}}}\text{ , where } \alpha = 0.5, \tag{10}$$

where $\mathcal{L}_{(\mathcal{S},g_{\mathcal{S}})}$ is the selective loss function of $\mathcal{S}$, as discussed in Section 2.2, and $\mathcal{L}_{h_{\mathcal{S}}}$ is the auxiliary loss function of $\mathcal{S}$ and are calculated as following:

$$\mathcal{L}_{(\mathcal{S},g_{\mathcal{S}})} \triangleq \hat{r}_\ell(\mathcal{S}, g_{\mathcal{S}} \mid S_k) + \lambda\Psi(c - \hat{\phi}(g_{\mathcal{S}} \mid S_k)),$$
$$\Psi(a) \triangleq \max(0, a)^2.$$

$$\mathcal{L}_{h_{\mathcal{S}}} = \hat{r}(h_{\mathcal{S}} \mid S_k) = \frac{1}{k}\sum_{i=1}^{k} \ell(h_{\mathcal{S}}(q_i), y_i).$$

3.4 Detection process in the test time

After having the $\mathcal{M}$ and the $\mathcal{S}$ classifiers trained, we can use them with the baseline classifiers $\mathcal{F}$ to detect the AEs in the inference/test time, As depicted in Fig. 2d. Specifically, the output of baseline model $P_c^{\mathcal{F}}$, the outputs of $\mathcal{M}$ block, $P_s^{m_j}$ and $P_c^{m_j}$, and the output of $\mathcal{S}$ block, $P_s^{\mathcal{S}}$ and $P_c^{\mathcal{S}}$, are used in the ensemble detection process. First of all, the following thresholds have to be identified:

– the confidence threshold value

$$th_c = \max(th_c^{\mathcal{S}}, th_c^{m_1}, th_c^{m_2}, ..., th_c^{m_N})$$

where $th_c^{m_j}$ is the confidence threshold for the selective AEs classifier m_j, and $th_c^{\mathcal{S}}$ is the confidence threshold for the $\mathcal{S}$ classifier.
- selective threshold $th_s^{m_j}$ for each selective AEs classifier m_j.
- selective threshold $th_s^{\mathcal{S}}$ for the $\mathcal{S}$ classifier.

Following the steps in [29], we select our thresholds using a subset of the clean test samples at a level when 10% (at most) of clean samples can be rejected by the ensemble detection. Once the thresholds are calculated we run the detection process as follows:

1. *Confidence detection:* is set to 1 if $max(P_c^{\mathcal{S}}) < th_c$ and is set to 0 otherwise, where 1 means adversarial input.
2. *Selective detection:* is set to 1 if $P_s^{\mathcal{S}} < th_s^{\mathcal{S}}$ or $P_s^{m_1} < th_s^{m_1}$ or ... or $P_s^{m_N} < th_s^{m_N}$ and is set to 0 otherwise.
3. *Mismatch detection:* is set to 1 if argmax $(P_c^{\mathcal{S}}) \neq$ argmax $(P_c^{\mathcal{F}})$ and is set to 0 otherwise.
4. *Ensemble detection:* The input sample is *adversarial* if it is detected in confidence, selective, or mismatch detection process.

4 Experimental settings

4.1 Datasets

The proposed prototype is evaluated on CNN models trained with two popular datasets; MNIST [45] and [46] CIFAR10.

MNIST is hand-written digit recognition dataset with 70000 images (60000 for training and 10000 for testing) and ten classes and CIFAR10 is an object recognition dataset with 60000 images (50000 for training and 10000 for testing) ten classes.

4.2 Baseline classifiers

For the baseline models, two CNN models are trained; one for MNIST and one for CIFAR10. For MNIST, we trained 6-layer CNN with 98.73% accuracy while for CIFAR10 we trained 8-layer CNN with 89.11% accuracy. The classifier's architectures for MNIST and CIFAR10 are shown in Table 1 and Table 2, respectively.

In order to evaluate the proposed prototypes against gray-box attacks, we consider that the adversaries know the training dataset and the model outputs and do not know the baseline model architectures. Hence, Table 3 and Table 4 show the two alternative architectures for MNIST and CIFAR10 classifiers. For MNIST, the classification accuracies are 98.37% and 98.69% for Model #2 and Model #3, respectively. While for CIFAR10, the classification accuracies are 86.93% and 88.38% for Model #2 and Model #3 respectively.

Table 1 MNIST baseline classifier architecture

Layer	Description
Conv2D + ReLU	32 filters (3×3)
Conv2D + ReLU + Max Pooling(2×2)	32 filters (3×3)
Conv2D + ReLU	64 filters (3×3)
Conv2D + ReLU + Max Pooling(2×2)	64 filters (3×3)
Dense + ReLU + Dropout ($p = 0.3$)	256 units
Dense + ReLU	256 units
Softmax	10 classes

4.3 SFAD Settings

As described in Section 3 and Fig. 2, we introduce here the implementation details for the detector components.

4.3.1 Selective AEs classifiers block

It consists of an autoencoder, up/down sampling, bottleneck, and noise layers. Each has the following architecture:

Autoencoder As shown in Fig. 3, let the input size be $Z \times w \times h$. In the encoding process, the number of 3×3-kernel filters are set to $Z/2$, $Z/4$, and $Z/16$, respectively. In the decoding process, the number of filters Z are symmetrically restored. Finally, to maintain the input samples characteristics that we have before autoencoding, the input is added/summed to the output of the autoencoder.

Up/down-sampling As shown in Fig. 4, let the input size be $Z \times w \times h$. The input size is doubled by bilinear up sampling in the first two consecutive layers and then restored by average pooling in the last two layers. Finally, to maintain the features before up/down sampling, the input is added to the output of up/down-sampling.

Bottleneck It is a three-convolutional layer module with kernels of size 1×1, 3×3, and 1×1. The architecture of the bottleneck layers are shown in Fig. 5. The number of the filters for each layer is 1024, 512, and 256.

Noise For this layer, the GaussianNoise layer model from Keras library is used with small standard variation of 0.05.

Dense layers A dense layer with 512 output is used followed by batch normalization and ReLU activation function.

SelectiveNet A dense layer with 512 outputs is used followed by batch normalization and ReLU activation function. After that, as original SelectiveNet's implementation suggests, a layer that divides the result of the previous layer

Table 2 CIFAR10 baseline classifier architecture

Layer	Description
Conv2D + BatchNorm + ReLU	64 filters (3×3)
Conv2D + BatchNorm + ReLU + Max Pooling(2×2) + Dropout ($p = 0.1$)	64 filters (3×3)
Conv2D + BatchNorm + ReLU	128 filters (3×3)
Conv2D + BatchNorm + ReLU + Max Pooling(2×2) + Dropout ($p = 0.2$)	128 filters (3×3)
Conv2D + BatchNorm + ReLU	256 filters (3×3)
Conv2D + BatchNorm + ReLU + Max Pooling(2×2) + Dropout ($p = 0.3$)	256 filters (3×3)
Conv2D + BatchNorm + ReLU + Max Pooling(2×2) + Dropout ($p = 0.4$)	512 filters (3×3)
Dense	512 units
Softmax	10 classes

Table 3 MNIST classifiers architectures for gray-box setting

Model	Layer	Description
Model #2	Conv2D + BatchNorm + ReLU	64 filters (3×3)
	Conv2D + BatchNorm + ReLU + Max Pooling(2×2) + Dropout ($p = 0.5$)	64 filters (3×3)
	Dense + BatchNorm + ReLU + Dropout ($p = 0.5$)	128 units
	Softmax	10 classes
Model #3	Conv2D + ReLU + Max Pooling(2×2)	32 filters (3×3)
	Conv2D + ReLU + Max Pooling(2×2)	64 filters (3×3)
	Dense + ReLU + Dropout ($p = 0.5$)	256 units
	Dense + ReLU	256 units
	Softmax	10 classes

Table 4 CIFAR10 classifiers architectures for gray-box setting

Model	Layer	Description
Model #2	Conv2D + BatchNorm + ReLU	32 filters (3×3)
	Conv2D + BatchNorm + ReLU + Max Pooling(2×2)	32 filters (3×3)
	Conv2D + BatchNorm + ReLU	64 filters (3×3)
	Conv2D + BatchNorm + ReLU + Max Pooling(2×2)	64 filters (3×3)
	Conv2D + BatchNorm + ReLU	128 filters (3×3)
	Conv2D + BatchNorm + ReLU + Max Pooling(2×2) + Dropout ($p = 0.4$)	128 filters (3×3)
	Dense + BatchNorm + ReLU + Dropout ($p = 0.5$)	512 units
	Softmax	10 classes
Model #3	Conv2D + ReLU	64 filters (3×3)
	Conv2D + ReLU + Max Pooling(2×2)	64 filters (3×3)
	Conv2D + ReLU	128 filters (3×3)
	Conv2D + ReLU + Max Pooling(2×2)	128 filters (3×3)
	Dense + ReLU + Dropout ($p = 0.5$)	256 filters (3×3)
	Dense + ReLU	256 filters (3×3)
	Softmax	10 classes

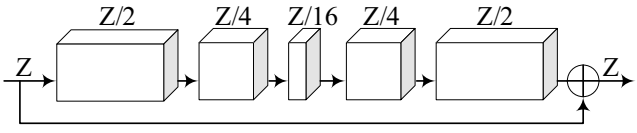


Fig. 3 Autoencoder architecture

by 10 is used as a normalization step. Finally, a dense layer of one output is used with sigmoid activation function. We set $\lambda = 32$, $c = 1$ for MNIST and $c = 0.9$ for CIFAR10, and coverage threshold to 0.995 for MNIST and 0.9 for CIFAR10. More details about selectiveNet hyper-parameters are found in [25].

4.3.2 Selective Knowledge Transfer block

It consists of one dense layer with 128 outputs followed by batch normalization and ReLU activation function. The selective task of the knowledge transfer block consists of a dense layer with 128 outputs followed by batch normalization and ReLU activation function. After that a normalisation layer that divides the result of the previous layer by 10 is used as recommended by the original implementation of SelectiveNet. Finally, a dense layer of one output is used with sigmoid activation function. We set $\lambda = 32$, $c = 1$ for MNIST and $c = 0.9$ for CIFAR10, and coverage threshold to 0.7 for MNIST and CIFAR10. More details about selectiveNet hyper-parameters are found in [25].

4.4 Threat model, attacks, and state-of-the-art detectors

4.4.1 Threat model

We follow one of the threat models presented in [47, 77]; Zero-Knowledge adversary threat model. It is assumed that the adversary has no knowledge that a detector is deployed and he generates the white-box attacks with the knowledge of the baseline classifier. For cases when an adversary has perfect or limited knowledge of the detector, we assume that the adversary's work will be so hard since SFAD adopts ensemble detection, and hence, we leave this as future work.

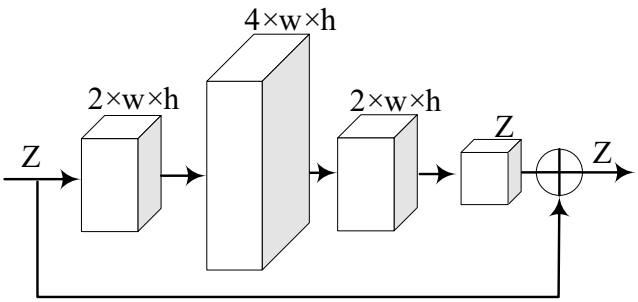


Fig. 4 Up/down-sampling architecture

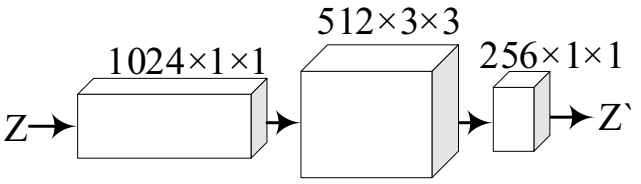


Fig. 5 Bottleneck architecture

Instead, we tested SFAD robustness with the recommended strong high confidence attack [31, 47], a variant of CW attack, that is rarely tested in other detectors.

4.4.2 Adversarial attacks

We test SFAD against different types of white and black box attacks. For the white box attacks, we use FGSM [7], PGD [14], CW [13], and DF attacks. While for the black-box attacks, we use TA [19], PA [18], and ST [17]. For the comparison with the state of the art algorithms, more black box attacks are considered like SA [78], and HopSkipJump [79] attacks. The attack settings are shown in Table 5.

Fast Gradient Sign Attack (FGSM) [7] It is a L_∞-norm attack and uses the model gradients to generate the AE. The sign of gradient for each pixel of the input x is used to build the AE x' as follows:

$$x' = x + \epsilon \, \text{sign}(\nabla_x \ell(x, y)), \text{ such that } x' \in [0, 1]^n \tag{11}$$

where ϵ is a parameter to control the perturbation amount such that $||x' - x||_\infty < \epsilon$.

Projected Gradient Descent (PGD) [14] It is the iterative version of the FGSM attack. PGD attack applies FGSM attack k times and starts from a random perturbation in L_p-ball around the input sample. It is expressed as:

$$\begin{aligned} &x'_{i+1} = x'_i + \alpha \, \text{sign}(\nabla_x \ell(x'_i, y)), \\ &\text{such that } x'_1 = x + rand(noise) \, , \\ &\quad x'_{i+1} \in [0, 1]^n \, , \text{ and } i = 1 \text{ to } k \end{aligned} \tag{12}$$

where α is the parameter to control the i^{th} iteration step size and it is $0 < \alpha < \epsilon$.

Carlini-Wagner (CW) [13] CW followed the optimization problem of the BFG [6] and replaced the loss function with an objective function:

$$g(x') = \max(\max_{i \neq t}(Z(x')_i) - Z(x')_t, -k), \tag{13}$$

where Z is the softmax function and k is the confidence parameter. Hence, CW solves the following optimization problem to build the AE:

$$\min_\delta ||\delta|| + c \, g(x'), \text{ such that } x' \in [0, 1]^n, \tag{14}$$

Table 5 Considered adversarial attacks and their parameters

Scenario	Attack	norm	Parameters
White box	FGSM	L_∞	$\epsilon \in \{0.05, 0.075, 0.1, 0.2, 0.4\}$
	PGD	L_∞	$\epsilon \in \{0.05, 0.075, 0.1, 0.2, 0.4\}$, $\epsilon_{step} = \epsilon/10$, *max. iterations*=100
	CW	L_∞	*confidence*=0, *max. iterations*=1000, *learning rate*=0.01
	DF	L_2	$\epsilon = 1e^{-6}$, *max. iterations*=100
Black box	SA	L_∞	$\epsilon = 16/255$, $p = 0.05$, *max. iterations*=300, *restarts*=1
	PA	L_0	L_0=10 for MNIST and find the minimum for the CIFAR, and textitmax. iterations=100
	TA	L_∞	*threshold*=find minimum, and *max. iterations*=100
	ST	-	-For MNIST, *translation*= 10 and *rotation*=60, and for CIFAR, *translation*= 8 and *rotation*=30
	HopSkipJump	L_∞	*type*=untargeted and unmasked, *iteration steps*=40, and *maximum evaluations*=100

where δ is the amount of perturbation and c is a regularisation parameter that we continuously search for to find minimum δ.

DF [12] Given a binary affine classifier $\mathcal{C} = \{x : f(x) = 0\}$, where $f(x) = w^T x + b$, DF attack defines the orthogonal projection of x_0 onto $\mathcal{C}$ as the minimal perturbation that is needed to change the classifier's decision, and it is calculated as $\delta_* = -\frac{f(x)}{||w||^2} w$. At each iteration, DF attack solves the following optimization problem

$$\begin{aligned} &\underset{\delta_i}{\text{argmin}}\ ||\delta_i||_2, \\ &\text{such that } f(x_i) + \nabla f(x_i)^T \delta_i = 0 \end{aligned} \tag{15}$$

and these perturbations are accumulated to get the final perturbation.

PA and TA [19] PA is a L_0-norm black box attack and uses the DEde (DE) [80] algorithm, to solve the optimization problem:

$$\max_{\delta} f(x + \delta) \text{ , such that } ||\delta||_0 \leq d \tag{16}$$

where d is a small number and equal to one in case of one-pixel. TA generalizes (16) to L_∞-norm attack to solve the optimization problem.

ST [17] ST applies translation and rotation changes to the input samples in order to fool the model and solves the optimization problem:

$$\max_{\delta u, \delta v, \theta} \ell(f(x'), y) \text{ , for } x' = T(x; \delta u, \delta v, \theta) \tag{17}$$

where $T, \delta u, \delta v$ and θ are, the transform function, x-coordinate translation, y-coordinate translation and angle rotation, respectively.

SA [78] In order to generate perturbation δ, SA, in each iteration, selects colored ϵ-bounded localized square shaped updates at random positions using random search strategy. Hence, it solves the optimization problem:

$$\min_{x' \in [0,1]^n} \ell(f(x'), y) \text{ , such that } ||\delta||_p \leq \epsilon \tag{18}$$

where $\ell(f(x'), y) = f_y(x') - \max_{k \neq y} f_k(x')$. $f_y(x')$ and $f_k(x')$ are the prediction probability scores of x' for y and k classes, respectively.

HopSkipJump attack [79] **(HSJA)** HopSkipJump is boundary-decision based black box attack that depends on estimating gradient-based direction. It starts from largely perturbed adversarial example δ and moves towards the clean input class boundary by minimizing the $||\delta||_2$.

4.4.3 Comparison with existing detectors

State-of-the-art supervised and unsupervised detectors are compared with SFAD. Supervised methods like KD+BU [22], LID [48], and RAID [64] are compared with SFAD. While unsupervised methods like FS [30], MagNet [65], NIC [35], and DNR [29] are also considered in the comparisons. A brief summary for each detector are demonstrated here:

KD+BU [22] It depends on building a binary classifier using two main features. The first one is the uncertainty features that are estimated using the Monte Carlo dropout technique [21]. The second feature depends on the kernel density estimation of each class in the training data.

rce [81] It depends on measuring the kernel density as in [22]. Instead of using the baseline classifier to measure the density functions, Pang et al. [81] measures the density functions using a more robust classifier that is trained using reverse cross entropy technique.

LID [48] Instead of measuring the kernel density, Ma et al. in [48] used Local Intrinsic Dimensionality (LID) to

calculate the distance distribution of the input sample to its neighbors.

RAID [64] It depends on measuring the differences in neuron activation values between clean and AEs inputs and then builds a binary classifier with these features.

FS [30] It depends on feature squeezing approach that transforms the input samples using squeezers. It uses color bit-depth reduction, local smoothing using median filter and non-local smoothing filter using non-local mean denoiser. To determine the adversarial status of an input, the distance between confidences of clean input and its squeezed version is calculated and compared with the threshold.

MagNet [65] First, it trains denoisers using clean training data. Then, it either 1) calculates the reconstruction error of the input and its denoised version, or 2) measures the distances between the predictions of an input sample and its denoised version to determine the adversarial status of an input.

NIC [35] It observes the behavior of clean training data only in the intermediate DL model layers. Specifically, it observes the provenance channel and the activation value distribution channels. The provenance channel describes the instability of activated neurons set in the next layer when small changes are present in the input sample, while the activation value distribution channel describes the changes with the activation values of a layer. For each individual layer, one-class classifiers (OCC) are built to model the in-distribution training data. A final one-class classifier that joins all one-class classifiers' outputs is used to determine the adversarial status of an input.

DNR [29] In this detector, Sotgiu et al. [29] uses the N-last representative layers outputs of the baseline classifiers to build N-SVM classifiers with RBF kernel. The output of these classifiers are then combined to build the joint SVM-RBF classifier. To determine the adversarial status of an input, the detector depends on checking the maximum confidence probability if it is less than a predefined threshold.

5 SFAD performance evaluation

In this section, we evaluate the performance of the SFAD prototype 1) against different types of attack scenarios and datasets, 2) against the strong high confidence attack, and then 3) we provide a comparison discussion with state-of-the-art detectors. As a reminder, we use only the last three representative layers ($N = 3$) to build three selective AEs classifiers since the aim is to prove the concept of the approach and if that is changed with the best combination, the detector accuracy will be enhanced accordingly.

5.1 Performance under white, black, and gray boxes attacks

5.1.1 Zero-Knowledge (of detectors) adversary white-box attacks

Table 6 shows the performance evaluation of the SFAD prototype for MNIST and CIFAR10 datasets. It also shows the baseline DNN prediction accuracy for the AEs in "Baseline DNN" row and for the not detected AEs in "prediction" row. The "Total" row is the total accuracy of ensemble detection and truly classified/predicted samples.

For MNIST dataset, the FGSM attacks with small epsilon ($\epsilon = 0.05$, 0.075, and 0.1) slightly fooled the baseline classifier and hence their feature space still inside or at the border as of training dataset. The detector shows its ability to reject those samples that are so close to the classes borders and achieves the accuracy of 99.96%, 99.88%, and 99.62% for $\epsilon = (0.05, 0.075,$ and, $0.1)$, respectively. Similar observation is noticed for PGD attacks with small ϵ values. For larger ϵ values, DF, and CW attacks, the AEs are highly able to fool the baseline classifier since adversaries are able to change the MNIST test samples' feature space to lie out of its corresponding class border and hence, for all tested attacks except the PGD, the model was able to catch them with accuracy above 98.65%. While the detector achieves 68.09% and 58.93% for PGD attacks with $\epsilon = (0.2,$ and, $0.4)$, respectively. Some PGD examples' feature space became indistinguishable from the trained samples feature space. That makes SFAD not able to catch all AEs and to enhance SFAD's performance, the best representative layers combination has to be used as input for the detector.

For CIFAR10 dataset, SFAD achieves comparable results with state-of-the-art methods for FGSM ($\epsilon = 0.1$, 0.2, and 0.4), DF, and CW attacks. While for FGSM ($\epsilon = 0.05$, and 0.075) and PGD attacks, the AEs have, to some extent, indistinguishable feature space than those the detector is trained with. In average, the model achieves accuracy of 65.2% for PGD attacks.

For both datasets, the effectiveness of selective, confidence, and mismatch detection is obvious, as shown in Table 7. The ability of the two modules to detect the AEs is increasing when the amount of the perturbations is increasing. When the amount of the perturbations increased in a way that makes the adversarial samples feature space indistinguishable from the training dataset, the ability of these modules to detect the AEs is decreasing.

Table 6 SFAD's performance accuracy (%) against white-box attacks(ϵ) on MNIST and CIFAR10 datasets at FP=10%

		FGSM (0.05)	FGSM (0.075)	FGSM (0.1)	FGSM (0.2)	FGSM (0.4)	FGSM (AVG)	PGD (0.05)	PGD (0.075)	PGD (0.1)	PGD (0.2)	PGD (0.4)	PGD (AVG)	DF	CW
	Baseline DNN	96.31	92.93	87.2	28.04	7.91	-	95.18	85.84	56.91	0	0	-	4.68	38.97
	Ensemble Detection	22.97	31.24	40.88	88.83	99.8	56.74	25.6	41.91	66.71	68.09	58.93	52.25	99.14	61.21
	Prediction	76.99	68.64	58.74	9.03	0	42.68	74.34	57.72	31.69	0	0	32.75	0.19	37.44
MNIST	**Total**	**99.96**	**99.88**	**99.62**	**97.86**	**99.8**	**99.42**	**99.94**	**99.63**	**98.4**	**68.09**	**58.93**	**85**	**99.33**	**98.65**
	Baseline DNN	14.09	13.44	12.25	10.5	9.75	-	0.43	0.28	0.22	0.16	0.17	-	4.79	20.95
	Ensemble Detection	72.07	81.84	88.42	99.41	100	88.34	57.57	63.59	66.67	68.77	68.74	65.07	88.45	69.93
	Prediction	6.94	3.28	1.39	0.02	0	2.33	0.34	0.13	0.07	0.06	0.04	0.13	1.35	20.09
CIFAR	**Total**	**79.01**	**85.12**	**89.81**	**99.43**	**100**	**90.67**	**57.91**	**63.72**	**66.74**	**68.83**	**68.78**	**65.2**	**89.8**	**90.02**

Prediction row is related to baseline DNN top-1 accuracy of not detected AEs. Total = SFAD's Ensemble Detection + Prediction

Table 7 SFAD's performance accuracy (%) of different detection processes against white-box attacks(ϵ) on MNIST and CIFAR10 datasets at FP=10%

		Detection process	FGSM (0.05)	FGSM (0.075)	FGSM (0.1)	FGSM (0.2)	FGSM (0.4)	FGSM (AVG)	PGD (0.05)	PGD (0.075)	PGD (0.1)	PGD (0.2)	PGD (0.4)	PGD (AVG)	DF	CW
MNIST	Selective	18.82	25.02	33.08	82.29	98.4	51.52	20.49	32.91	51.81	58.07	48.17	42.29	95.7	43.71	
	Confidence	8.66	13.74	20.75	74.18	98.66	43.2	10.31	21	42.73	54.14	47.41	35.12	94.96	42.78	
	Mismatch	3.07	5.66	9.77	43.43	63.93	25.17	3.85	10.5	28.61	26.86	18.69	17.7	59.47	44.99	
	Ensemble	22.97	31.24	40.88	88.83	99.8	56.74	25.6	41.91	66.71	68.09	58.93	52.25	99.14	61.21	
CIFAR	Selective	41.31	46.8	53.78	71.34	9.83	44.61	33.6	33.18	30.52	22.66	18.65	27.72	39.06	37.41	
	Confidence	69.3	80.61	87.35	99.09	99.99	87.27	43.33	54.69	60.86	66.83	67.68	58.68	85.76	65.47	
	Mismatch	25.49	35.15	42.88	44.41	59.66	41.52	0	0	0.02	0.11	1.32	0.29	37.5	34.42	
	Ensemble	72.07	81.84	88.42	99.41	100	88.34	57.57	63.59	66.67	68.77	68.74	65.07	88.45	69.93	

Table 8 SFAD's performance accuracy (%) against black-box attacks on MNIST and CIFAR10 datasets at FP=10%

Dataset	Attack	Baseline DNN	SFAD (ours)		
			Ensemble prediction	Prediction	Total
MNIST	Threshold Attack	77.61	85.62	14.31	99.93
	Pixel Attack	74.57	85.76	14.18	99.94
	Spatial Transformation	22.04	94.74	2.85	97.59
CIFAR	Threshold Attack	11.29	92.62	1.35	93.97
	Pixel Attack	11.35	92.77	1.39	94.16
	Spatial Transformation	52.58	72.54	24.03	96.57

Prediction column is related to baseline DNN top-1 accuracy of not detected AEs. Total = SFAD's Ensemble Detection + Prediction

5.1.2 Black-box attacks

Table 8 shows SFAD prototype's detection accuracy against the TA [19], PA [18], and ST [17] attacks on MNIST and CIFAR10 datasets. The detector is able to catch the AEs with very high accuracy, higher than 97.56% and 93.97% for MNIST and CIFAR10, respectively. It is clear that the selective, confidence, and mismatch modules complement each other. The black-box attacks significantly change the samples features that facilitate the confidence module detection process. While the ability of selective module is limited for TA and PA attacks since these attacks change one or more pixels within a threshold that is in a variation of the input sample and yield AEs that are so close to clean samples. Similar to white box attacks, the effectiveness of selective, confidence, and mismatch detection is obvious for the both datasets as shown in Table 9.

5.1.3 Gray-box attacks

Gray-box scenario assumes that the adversary has only knowledge about the model training data and the output of the DNN model. Hence, we trained two models as substitution models named Model#2 and Model#3 for MNIST and CIFAR10 as shown in Tables 3 and 4, respectively. Then, white-box based AEs are generated using the substitution models. The SFAD prototype is then tested against these AEs. For both datasets, it is shown in Tables 10 and 11 that the perturbations properties generated from one model are transferred to the tested model, Model#1. For MNIST, see Table 10, SFAD prediction rate is much better for PGD attacks and the prediction rate for other attacks is comparable with the prediction rate of AEs generated from Model#1. For CIFAR10, see Table 11, the prediction rate for CW and DF attacks is higher than those attacks that are generated using Model#1, while the prediction rate for FGSM is comparable with the prediction rate for FGSM attacks that are generated using Mode1#1. Unlike other attacks, the PGD attacks transferable properties sound to be much stronger and have different feature space, compared to feature space of AEs that are generated from Model#1. This reduces the ability of the detector to catch such attacks.

5.2 Robustness against high confidence attack

In [31], ten defenses and detectors were broken using Backward Pass Differentiable Approximation (BPDA), Expectation Over Transformation (EOT), and High Confidence Attack (HCA). BPDA, and EOT are appropriate for defense techniques, while HCA is used to fail detectors. HCA is a variant of CW attack and generates adversarial examples with high confidence level. In [31], LID were broken using HCA. In his experiment, we generate AEs using HCA with $\epsilon = 0.3125$ for MNIST and $\epsilon = 0.031$ for CIFAR10. The results show that SFAD is fully robust on MNIST against HCA and partially robust (57.76%) on CIFAR10. Our analysis finds that the confidence and selective detection methods are effective to detect AEs. In case the confidence level of the attack is increased, SFAD can be fine-tuned by selecting the proper layer outputs to build the selective AE classifiers.

All the experiments that are conducted in this work are tested under zero knowledge of the detector. We assume that the adversary's work is very hard for building an adaptive attack to fool SFAD since it ensembles three detection methods. Despite that, SFAD performance will drop when the adversary is able to craft customized perturbations to fool both the baseline classifier and the ensemble detector.

5.3 Comparisons with the state-of-the-art detectors

In this subsection we build a comparison with different types of supervised and unsupervised detectors using the detectors benchmark[1] [82] and the results are shown in Table 12. We compare the average FGSM and PGD results.

[1]The source code is available in https://github.com/aldahdooh/detectors_review

Table 9 SFAD's performance accuracy (%) of different detection processes against black-box attacks on MNIST and CIFAR10 datasets at FP=10%

Dataset	Attack	SFAD (ours)			
		Selective Detection	Confidence Detection	Mismatch Detection	Ensemble Prediction
MNIST	Threshold Attack	24.36	85.48	42.37	85.62
	Pixel Attack	24.65	85.57	42.88	85.76
	Spatial Transformation	86.7	80.95	34.71	94.74
CIFAR	Threshold Attack	12.69	92.14	37.11	92.62
	Pixel Attack	12.48	92.4	37.02	92.77
	Spatial Transformation	44.64	68.16	32.44	72.54

Table 10 SFAD's performance accuracy (%) against gray-box attacks(ϵ) on MNIST dataset at FP=10%

Attack (ϵ)	Model#1	Model#2			Model#3		
	Total	Ensemble Detection	Prediction	Total	Ensemble Detection	Prediction	Total
FGSM(0.05)	99.96	13.86	86.14	100	15.28	84.72	100
FGSM(0.075)	99.88	16.23	83.77	100	18.84	81.14	99.98
FGSM(0.1)	99.62	19.59	80.41	100	23.66	76.29	99.95
FGSM(0.2)	97.86	51.49	48.25	99.74	63.32	35.59	98.91
FGSM(0.4)	99.8	99.41	0.41	99.82	99.89	0	99.89
PGD(0.05)	99.94	13.57	86.43	100	15.49	84.49	99.98
PGD(0.075)	99.63	15.01	84.99	100	19.77	80.21	99.98
PGD(0.1)	98.4	18.53	81.47	100	25.73	74.21	99.94
PGD(0.2)	68.09	54.9	44.44	99.34	73.26	23.85	97.11
PGD(0.4)	58.93	91.1	0.93	92.03	82.78	0.2	82.98
DF	99.33	92.25	7.37	99.62	96.03	2.59	98.62
CW	98.65	25.16	74.82	99.98	38.57	61.33	99.9

Prediction column is related to baseline DNN top-1 accuracy of not detected AEs. Total = SFAD's Ensemble Detection + Prediction

Table 11 SFAD's performance accuracy against gray-box attacks(ϵ) on CIFAR10 dataset at FP=10%

Attack (ϵ)	Model#1	Model#2			Model#3		
	Total	Ensemble Detection	Prediction	Total	Ensemble Detection	Prediction	Total
FGSM(0.05)	79.01	74.09	8.56	82.65	76.43	7.9	84.33
FGSM(0.075)	85.12	79.68	4.26	83.94	80.25	4.17	84.42
FGSM(0.1)	89.81	87.48	2.03	89.51	83.69	2.55	86.24
FGSM(0.2)	99.43	99.66	0.03	99.69	96.9	0.26	97.16
FGSM(0.4)	100	100	0	100	100	0	100
PGD(0.05)	57.91	20.7	5.62	26.32	18.56	4.44	23
PGD(0.075)	63.72	13.18	5.13	18.31	10.36	4.31	14.67
PGD(0.1)	66.74	12.6	5.07	17.67	9.6	4.3	13.9
PGD(0.2)	68.83	19.64	4.9	24.54	19.35	3.92	23.27
PGD(0.4)	68.78	31.76	4.35	36.11	54.21	2.37	56.58
DF	89.8	81.2	16.17	97.37	84.28	11.74	96.02
CW	90.02	45.35	53.29	98.64	55.06	42.18	97.24

Prediction column is related to baseline DNN top-1 accuracy of not detected AEs. Total = SFAD's Ensemble Detection + Prediction

Table 12 Detection accuracies for the state-of-the-art detectors against white-box and black-box attacks

Dataset	Detector	Attacks								
		FPR	White box					Black box		
			FGSM	PGD	CW	HCA	DF	SA	HSJA	STA
MNIST	KD+BU[†]	1.76	62.69	52.52	42.77	59.26	48.97	53.53	61.82	47.94
	LID[†]	0.81	77.46	77.03	64.43	84.51	93.3	42.78	61.52	**93.81**
	FS*	5.27	97.96	**97.19**	**98.41**	**99.99**	66.96	**99.96**	**99.98**	77.49
	MagNet*	0.20	**100.00**	**100.00**	40.56	**100.00**	**96.99**	**99.93**	98.32	1.61
	NIC*	10.12	**100.00**	**100.00**	**100**	**100.00**	**100**	**99.68**	**100**	**99.83**
	DNR*	10.01	79.67	59.21	57.98	89.90	95.6	81.27	59.98	88
	SFAD* (ours)	10.79	98.66	81.83	**98.24**	99.36	**99.58**	98.85	**99.91**	**97.61**
CIFAR10	KD+BU[†]	3.57	34.13	**96.08**	21.12	33.49	54.02	0	28.03	44.15
	LID[†]	7.30	**76.15**	**96.81**	**64.52**	**55.03**	**63.57**	**85.76**	**88.34**	**94.23**
	FS*	5.07	32.50	4.20	56.18	**56.31**	39.18	17.82	**84.16**	22.46
	MagNet*	0.77	34.61	0.62	13.23	0.53	57.33	**94.04**	0.58	2.32
	NIC*	10.08	**63.15**	**100.00**	**61.68**	**73.29**	**84.91**	61.88	67.53	48.77
	DNR*	10.01	30.23	18.23	44.15	29.85	30.2	52.86	38.81	**56.2**
	SFAD* (ours)	10.90	**80.14**	41.2	**87.68**	45.85	**89.57**	**93.91**	**95.57**	**92.9**

footnotetext[]Top 3 are colored with red, blue, and green respectively. All detectors that are mentioned here are re-implemented

[1]Supervised detector, * Unsupervised detector

For fair comparisons, ϵ values of 0.125, 0.25, and 0.3125 are set for MNIST dataset, while for CIFAR10 are set to 0.03 and 0.06. Moreover, the supervised detectors are trained and tested separately against each adversarial attack algorithm. As discussed in Section 6.4, rejection/false positive rates of SFAD can be decreased with small compromise in the performance.

KD+BU [22] KD+BU detector is a combination of kernel density and Bayesian uncertainty based classifiers. For both datasets, the results show that SFAD outperforms KD+BU detector against all tested attacks except for PGD attacks in CIFAR10 dataset. In fact, KD+BU needs not noisy clean and adversarial images to accurately train the detector to identify the boundaries between clean and adversarial inputs.

RCE [2]. Compared to KD [22], RCE achieves better performance since its classifier yields latent representations that better distinguish AEs from normal examples. For the both datasets, MNIST and CIFAR10, reverse cross entropy (RCE) yields better area under the curve scores than KD, while it shows limited performance against the basic iterative method (BIM) [11] and the HCA.

[2] Detector is compared with the results that are reported in the original paper [81]

LID [48] SFAD outperforms LID in both datasets and the tested attacks except for PGD attacks on CIFAR10. LID achieves better false positive rate compared to SFAD but it fails against High Confidence Attack as reported in [47]. When LID is trained for the HCA attacks, it achieves better results than in [47]. Our approach provides full and partial robustness against HCA for MNIST and CIFAR10, respectively. Similar to KD+BU, LID needs not noisy clean and adversarial images to accurately train the detector to identify the boundaries between clean and adversarial inputs.

RAID[3] [64]. For MNIST dataset, RAID achieves higher detection rate for PGD attacks ($\epsilon = 0.3$) and higher detection rate against FGSM and PGD attacks for CIFAR10 while our approach improved the performance against CW and DF attacks. Besides, RAID has a better false positive rate for MNIST only. RAID trains clean and adversarial inputs to identify differences in neuron activation between clean and adversarial samples. Hence, it requires a huge knowledge of attacks and its variants to enhance its performance.

FS [30] As stated in [30], FS requires high quality squeezers for different baseline networks and it was shown that FS is not performing well against tested attacks on CIFAR10

dataset, while our approach generalizes better than FS at the expense of higher false positive rate.

MagNet [65] Results reported on Table 12 is for the detection process of MagNet and defense process of MagNet is not considered. For MNIST, comparable results are achieved by our approach except for CW and ST attacks where SFAD achieves better performance. For CIFAR, our approach outperforms MagNet against the tested attacks. Since MagNet is a denoiser-based detector, it is not guaranteed that the denoisers will remove all the noise and have highly denoised inputs that respect the target threshold. This applies specifically to L_0 and L_2 attacks. On the contrary, our approach relies on confidence value changes that the AEs will cause which makes our approach able to identify AEs. Although MagNet yields to a less false positive rate, it was shown in [32] that MagNet can be broken by different strategies.

NIC [35] NIC is the state-of-the-art detector that achieves better performance, in general, against white box attacks compared to other detectors, while our approach achieves better performance against tested black box attacks. Unlike the proposed approach, other works reported that the NIC's baseline detectors are not consistent [33], increase the model parameters overhead [34], are time consuming [35], and have latency in the inference time [36].

DNR [29] DNR adopted confidence-based detectors and is close to our approach, but we include the feature processing and selective modules components. The reported results show that our approach outperforms DNR at the same false positive rates for MNIST and CIFAR10 datasets.

Other performance comparison: SFAD has middle complexity level due to classifiers training times, and has no inference time latency, but it has a compromise on overhead due to classifiers parameters saving. Compared to other detectors, SFAD introduces shallow networks hence, compared to NIC, DNR, and LID, our detector has much less complexity. Besides, it works in parallel to the baseline classifier and no latency is provided compared to FS and NIC. Finally, like NIC and DNR, SFAD has to pay a little price in terms of overhead compared to MagNet, FS and LID.

6 Other experimental results and discussion

In this section, more performance analysis is discussed in order to validate the SFAD prototype. First, we evaluate SFAD against successful attacks only[3]. Then, the proposed approach is tested with different N settings. Moreover, in order to emphasize the advantages of SFAD's feature processing components, we provide an ablation study for each component. Finally, performance results on different rejection rates, i.e. false positive rates are shown.

6.1 Performance on successful attacks only

Table 13 shows the detection rate against the AEs that fooled the baseline DNN classifier only under white-box and black-box scenarios. For MNIST, in general, comparable results with the state-of-the-art detectors are achieved for all tested white and black boxes attacks ($> 96.91\%$) except for the PGD attacks (83.88%).

For both datasets, the impact of selective, confidence, and mismatch detection modules are obvious. The ability of the modules to detect the AEs is increasing when the amount of the perturbations is increasing. When the amount of the perturbations increases in a way that makes the adversarial samples feature space indistinguishable from the training dataset, the ability of these modules to detect the AEs decreases. Mismatch detection shows a high impact in the detection process of AEs except for PGD attacks. Once the amount of crafted perturbation becomes high, the performance of mismatch detection decreases. That's because the detector classifiers' and the baseline DNN classifier's behavior will be inconsistent for highly degraded inputs.

6.2 Results with *N* last layer(s) output(s)

Results shown in Fig. 6 emphasize the conclusion in [29] that recommends to use more than one layer from the last layers of the baseline DNN classifier to be used in detection techniques. For MNIST dataset, the benefit of using more than one layer appears in detecting PGD ($\epsilon = 0.2$, and 0.4), TA, and PA attacks, while it appears in all tested attacks on CIFAR10 dataset. It means that low-/and medium-level hidden layers hold features that will be triggered when small perturbations are added to input samples.

6.3 Ablation study

In this section, we emphasize the advantages of SFAD's feature processing components including noise, autoencoder, up/down sampling, and bottleneck blocks. Tables 14 and 15 show the performance results for each block once when it is

[3]The AEs that are able to fool a model are called successful AEs, otherwise, are called failed or unsuccessful AEs

Table 13 Detection modules' accuracies (%) against *successful* white-box and black-box attacks(ϵ) on MNIST and CIFAR10 datasets at FP=10%

		White-box attacks													Black-box attacks		
		FGSM (0.05)	FGSM (0.075)	FGSM (0.1)	FGSM (0.2)	FGSM (0.4)	FGSM (AVG)	PGD (0.05)	PGD (0.075)	PGD (0.1)	PGD (0.2)	PGD (0.4)	PGD (AVG)	DF	CW	TA	PA	ST
MNIST	Selective Detection	59.89	60.74	64.56	89.4	98.33	**74.58**	57.35	64.23	68.05	58.07	48.17	**59.17**	95.78	69.16	25.96	28.04	87.9
	Confidence Detection	60.99	60.6	63.29	88.56	98.57	**74.4**	59.45	63.66	71.16	54.14	47.41	**59.16**	95	69.96	99.19	99.08	88.1
	Mismatch Detection	77.2	74.21	71.28	59.03	62.95	**68.93**	74.16	70.17	64.25	26.86	18.69	**50.83**	58.87	73.73	86.84	86.82	40.96
	Ensemble Detection	98.9	98.42	97.07	97.04	99.78	**98.24**	98.74	97.35	96.31	68.09	58.93	**83.88**	99.3	97.8	99.73	99.8	96.91
CIFAR	Selective Detection	44.7	49.05	54.7	70.16	10.2	**45.76**	33.74	33.25	30.6	22.68	18.68	**27.79**	39.12	47.32	10.92	10.84	60.08
	Confidence Detection	72.57	81.45	87.35	99.03	99.99	**88.08**	43.44	54.69	60.84	66.83	67.68	**58.7**	86.53	81.74	92.78	93.05	87.96
	Mismatch Detection	27.62	36.19	42.58	42.67	58.52	**41.52**	0	0	0.02	0.11	1.33	**0.29**	37.79	43.54	39.15	38.92	54.35
	Ensemble Detection	75.57	82.82	88.4	99.36	100	**89.23**	57.74	63.62	66.66	68.71	68.74	**65.09**	89.33	87.38	93.21	93.42	92.76

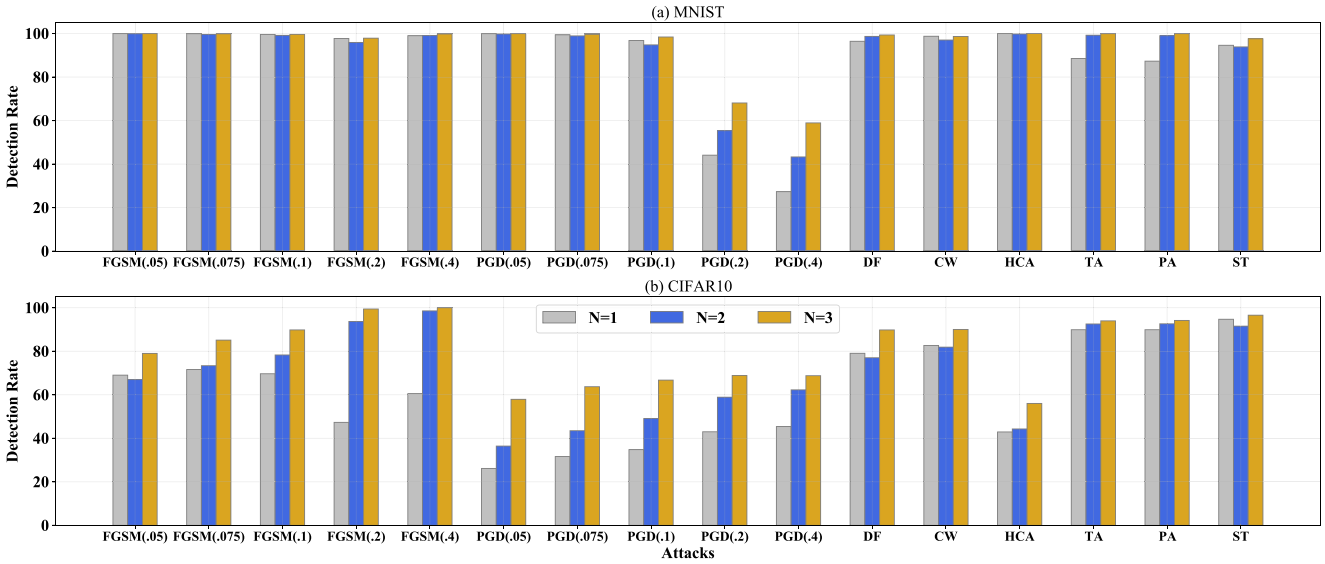


Fig. 6 Total model performance accuracy (%) for black and white box scenarios on MINIST and CIFAR10 datasets at FP=10% with different *N* selective AEs classifiers settings

present alone and another time when it is absent for MNIST and CIFAR10 datasets. In all settings, the selectiveNet is present in the selective AE classifiers and in the selective knowledge transfer classifier.

Only NN When all processing blocks are absent, the MNIST results show the ability to detect FGSM, PGD of small ϵ values, and CW attacks slightly better than the proposed approach. While the proposed approach yields better results for DF and PGD of high ϵ values. Since CIFAR10 dataset is different from MNIST and has different characteristics, the only NN component did not yield better results against FGSM of high ϵ values, PGD, CW, and DF attacks.

Noise When only the noise block is used, the model achieves comparable results to SFAD except against PGD attacks. When we remove the noise block, the performance of SFAD is reduced especially against PGD attacks for MNIST and CIFAR10 datasets. The noise block helps the detector to better distinguish the feature space of clean input images from those features of AEs.

Autoencoder Autoencoder block shows a substantial impact in the proposed approach. As discussed in Section 3.2.1, if the autoencoder couldn't reconstruct its input, different feature space might be generated for the input signal which let SFAD able to detect the AEs. For MNIST dataset, the

Table 14 Ablation performance (%) on white-box scenarios for MNIST dataset

Attack/Model	Baseline DNN	NN	Only noise	Only auto encoder	Only up/down sampling	Only bottle-neck	No noise	No auto encoder	No up/down sam-pling	No bottleneck	Proposed
FGSM(0.05)	96.31	**100**	99.97	**100**	**100**	99.92	99.95	99.96	99.95	99.97	99.96
FGSM(0.075)	92.93	**99.98**	99.94	99.96	99.97	99.81	99.76	99.9	99.8	99.92	99.88
FGSM(0.1)	87.2	99.9	**99.93**	99.9	**99.93**	99.69	99.49	99.79	99.51	99.86	99.62
FGSM(0.2)	28.04	**99.61**	99.44	98.97	98.78	96.71	96.37	96.96	97.79	98.79	97.86
FGSM(0.4)	7.91	98.98	98.37	98.65	96.14	97.85	93.73	**99.87**	98.81	97.01	99.8
PGD(0.05)	95.18	**100**	99.95	99.97	99.98	99.88	99.87	99.93	99.91	99.95	99.94
PGD(0.075)	85.84	99.83	**99.86**	99.84	**99.86**	99.58	99.43	99.73	99.57	99.76	99.63
PGD(0.1)	56.91	**99.35**	99.2	99.06	99.09	98.09	97.65	98.43	97.7	98.92	98.4
PGD(0.2)	0	60.06	59.33	66.24	53.73	66.18	57.9	61.39	63.44	63.93	**68.09**
PGD(0.4)	0	43.8	42.61	52.57	40.97	46.11	45.11	43.39	46.57	47.71	**58.93**
DF	4.68	97.71	97.35	98.44	95.15	98.51	96.88	99	98.83	98.62	**99.33**
CW	38.97	**99.71**	99.66	99.52	99.6	98.36	97.73	98.79	98.57	99.19	98.65

Table 15 Ablation performance (%) on white-box scenarios for CIFAR10 dataset

Attack/Model	Baseline DNN	NN	Only noise	Only auto encoder	Only up/down sampling	Only bottle-neck	No noise	No auto encoder	No up/down sam-pling	No bottleneck	Proposed
FGSM(0.05)	14.09	**80.02**	79.55	73	80.01	79.68	78.62	75.81	75.7	78.55	79.01
FGSM(0.075)	13.44	**87.9**	85.21	80.44	85.84	86.61	86.2	84.77	81.3	85	85.12
FGSM(0.1)	12.25	**91.37**	87.26	84.64	87.71	91	91.27	90.75	85.56	88.26	89.81
FGSM(0.2)	10.5	85.52	78.89	95.13	78.73	97.79	**99.48**	98.19	95.69	88.92	99.43
FGSM(0.4)	9.75	84.39	84.94	99.71	79.32	**100**	**100**	99.98	**100**	99.85	**100**
PGD(0.05)	0.43	0.42	0.43	4.21	0.42	1.08	36.61	7.8	48.34	12.58	**57.91**
PGD(0.075)	0.28	0.27	0.28	11.64	0.27	2.76	47.19	9.65	58.32	12.67	**63.72**
PGD(0.1)	0.22	0.22	0.22	17.72	0.22	5.22	52.68	12.09	61.49	13.42	**66.74**
PGD(0.2)	0.16	0.15	0.15	27.83	0.16	14.7	58.23	18.86	64.31	16.61	**68.83**
PGD(0.4)	0.17	0.17	0.17	33.74	0.16	22.75	60.23	23.9	65.4	18.21	**68.78**
DF	4.79	83.67	84.84	83.71	82.7	89.11	88.32	87.64	87.96	88.09	**89.8**
CW	20.95	88.98	89.05	87.03	88.33	89.46	88.63	87.7	88.14	89.26	**90.02**

autoencoder block enhanced the performance results compared to only NN model against PGD of higher ϵ values, while the performance is reduced when the autoencoder block is removed from the proposed approach. On the other hand, for CIFAR10 dataset, when only the autoencoder is present, the performance results are much better against FGSM of high ϵ values, PGD, CW, and DF attacks when it is compared to only NN. The performance is reduced when it is removed from the proposed approach against PGD attacks.

Up/down-sampling Unlike other processing blocks, up/ down sampling block yields less performance results against FGSM attacks and yields comparable results against other attacks compared to only NN model. That's because the up/ down-sampling restores the global information of the input signal by the average pooling process. On the other hand, removing the sampling block from the proposed approach reduces the performance results especially for the CIFAR10 dataset.

Bottleneck Like autoencoder block, the bottleneck block shows its ability to distinguish input signal characteristics especially in the proposed shallow classifiers (the selective AEs classifiers). Compared to only NN model, the only bottleneck model enhanced the performance results against FGSM of high ϵ values, PGD, CW, and DF attacks for CIFAR10 dataset and enhanced the performance results against PGD of high ϵ values attacks for MNIST. Besides, the performance of the proposed approach is significantly decreased for CIFAR10 dataset when the bottleneck block is removed.

6.4 Performance with different rejection rates (False positive (FP))

In this subsection we show the performance results of the proposed approach when thresholds are set to reject less than 10% for MNIST as shown in Fig. 7. Results show that

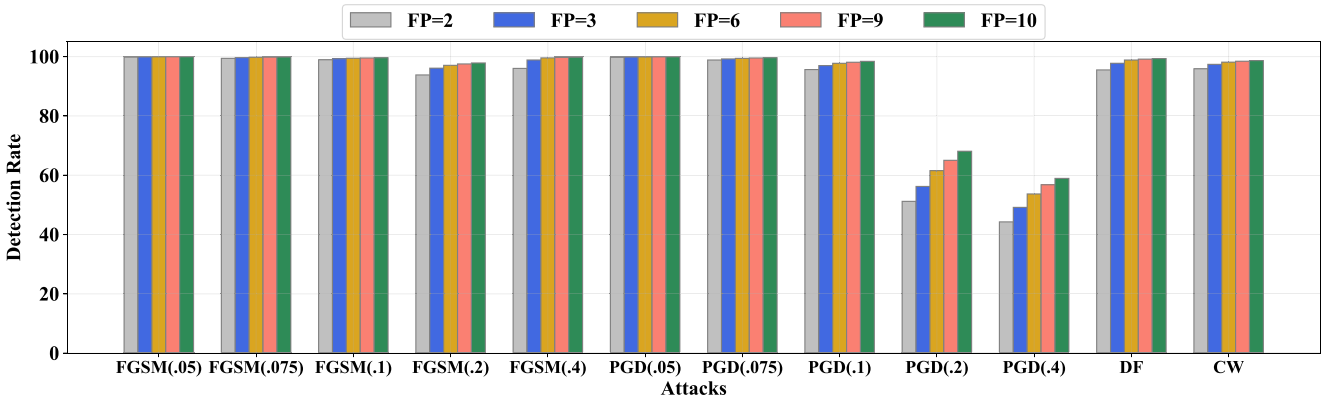


Fig. 7 Performance comparisons between different False Positive (FP) rates and FP=10% of SFAD for white-box attacks on MNIST dataset

an acceptable performance can be achieved if the thresholds are set to less than 10%. For instance, when the false positive rate is set to be 2%, results against PGD ($\epsilon = 0.2$, , and 0.4) attacks are significantly decreased because of the selective detection. In all other tested attacks, the difference is up to 4% and 1.76% when FP=2% and 3%, respectively.

7 Conclusion

In this work, we have proposed a novel unsupervised and ensemble mechanism, namely SFAD, to detect adversarial attacks. SFAD handled the N-last layers outputs of the baseline DNN classifier to identify AEs. It built N selective AEs classifiers that each took one layer output of the baseline classifier as input and then processed the input using autoencoder, up/down sampling, bottleneck, and additive noise blocks. Then, these feature-based classifiers were optimized in the SelectiveNet model to estimate the model's uncertainties and confidences. The confidence values of these classifiers were then distilled as input to the selective knowledge transfer classifier to build the last classifier. Selective and confidence thresholds were set to identify the adversarial inputs. Selective, confidence, and mismatch modules are jointly working to enhance the detection accuracy. We showed that the model is consistent and is able to detect tested attacks. Moreover, the model is robust in different attack scenarios; white, black, and gray boxes attacks. This robustness, with the advantage that the model does not require any knowledge of adversarial attacks, will lead to better generalization. The main limitation of the model is that the best combination of N needs to be identified to enhance the detection accuracy and to reduce the false positive rate.

Acknowledgements The project is funded by both Région Bretagne (Brittany region), France, and direction générale de l'armement (DGA).

References

1. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
2. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings, San Diego
3. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99
4. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
5. Shen D, Wu G, Suk H-I (2017) Deep learning in medical image analysis. Ann Rev Biomed Eng 19:221–248
6. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow IJ, Fergus R (2014) Intriguing properties of neural networks. In: Bengio Y, LeCun Y (eds) 2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings, Banff
7. Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings, San Diego
8. Guo W, Mu D, Xu J, Su P, Wang G, Xing X (2018) Lemna: Explaining deep learning based security applications. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp 364–379
9. Akhtar N, Mian A (2018) Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access 6:14410–14430
10. Hao-Chen HXYM, Deb LD, Anil HLJ-LT, Jain K (2020) Adversarial attacks and defenses in images, graphs and text: A review. Int J Autom Comput 17(2):151–178
11. Kurakin A, Goodfellow I, Bengio S (2017) Adversarial examples in the physical world. ICLR Workshop
12. Moosavi-Dezfooli S-M, Fawzi A, Frossard P (2016) Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2574–2582
13. Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp). IEEE, pp 39–57
14. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2018) Towards deep learning models resistant to adversarial attacks. In: 6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings. OpenReview.net, Vancouver
15. Papernot N, McDaniel PD, Goodfellow IJ (2016) Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. CoRR arXiv:1605.07277
16. Chen P-Y, Zhang H, Sharma Y, Yi J, Hsieh C-J (2017) Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp 15–26
17. Engstrom L, Tran B, Tsipras D, Schmidt L, Madry A (2019) Exploring the landscape of spatial robustness. In: International Conference on Machine Learning, pp 1802–1811
18. Su J, Vargas DV, Sakurai K (2019) One pixel attack for fooling deep neural networks. IEEE Trans Evol Comput 23(5):828–841
19. Kotyan S, Vasconcellos Vargas D (2019) Adversarial robustness assessment: Why both l_0 and l_∞ attacks are necessary, pp arXiv–1906
20. Gal Y, Ghahramani Z (2016) Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: International Conference on Machine Learning. PMLR, pp 1050–1059
21. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
22. Feinman R, Curtin RR, Shintre S, Gardner AB (2017) Detecting adversarial samples from artifacts. CoRR arXiv:1703.00410
23. Smith L, Gal Y (2018) Understanding measures of uncertainty for adversarial example detection. In: Globerson A, Silva R (eds) Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018. AUAI Press, Monterey, pp 560–569
24. Sheikholeslami F, Jain S, Giannakis GB (2020) Minimum uncertainty based detection of adversaries in deep neural

networks. In: Information Theory and Applications Workshop, ITA 2020. IEEE, San Diego, pp 1–16
25. Geifman Y, El-Yaniv R (2019) Selectivenet: A deep neural network with an integrated reject option. CoRR arXiv:1901.09192
26. Hendrycks D, Gimpel K (2017) A baseline for detecting misclassified and out-of-distribution examples in neural networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net
27. Aigrain J, Detyniecki M (2019) Detecting adversarial examples and other misclassifications in neural networks by introspection. CoRR arXiv:1905.09186
28. Monteiro J, Albuquerque I, Akhtar Z, Falk TH (2019) Generalizable adversarial examples detection based on bi-model decision mismatch. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). IEEE, pp 2839–2844
29. Sotgiu A, Demontis A, Melis M, Biggio B, Fumera G, Feng X, Roli F (2020) Deep neural rejection against adversarial examples. EURASIP J Inf Secur 2020:1–10
30. Xu W, Evans D, Qi Y (2018) Feature squeezing: Detecting adversarial examples in deep neural networks. In: 25th Annual Network and Distributed System Security Symposium, NDSS 2018. The Internet Society, San Diego
31. Athalye A, Carlini N, Wagner DA (2018) Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: Dy JG, Krause A (eds) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan. Proceedings of Machine Learning Research, vol 80. PMLR, Stockholm, pp 274–283
32. Carlini N, Wagner DA (2017) Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. CoRR arXiv:1711.08478
33. Bulusu S, Kailkhura B, Li B, Varshney PK, Song D (2020) Anomalous example detection in deep learning: A survey. IEEE Access 8:132330–132347
34. Lust J, Condurache AP (2020) Gran: An efficient gradient-norm based detector for adversarial and misclassified examples. In: 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, Bruges, pp 7–12
35. Ma S, Liu Y (2019) Nic: Detecting adversarial samples with neural network invariant checking. In: Proceedings of the 26th Network and Distributed System Security Symposium (NDSS 2019)
36. Gao Y, Doan BG, Zhang Z, Ma S, Zhang J, Fu A, Nepal S, Kim H (2020) Backdoor attacks and countermeasures on deep learning: A comprehensive review. CoRR aRxiv:2007.10760
37. Melis M, Demontis A, Biggio B, Brown G, Fumera G, Roli F (2017) Is deep learning safe for robot vision? adversarial examples against the icub humanoid. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp 751–759
38. Lu J, Issaranon T, Forsyth D (2017) Safetynet: Detecting and rejecting adversarial examples robustly. In: Proceedings of the IEEE International Conference on Computer Vision, pp 446–454
39. Wang F, Jiang M, Qian C, Yang S, Li C, Zhang H, Wang X, Tang X (2017) Residual attention network for image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3156–3164
40. Liu S, Johns E, Davison AJ (2019) End-to-end multi-task learning with attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1871–1880
41. Lecuyer M, Atlidakis V, Geambasu R, Hsu D, Jana S (2019) Certified robustness to adversarial examples with differential privacy. In: 2019 IEEE Symposium on Security and Privacy (SP). IEEE, pp 656–672
42. Liu X, Cheng M, Zhang H, Hsieh C-J (2018) Towards robust neural networks via random self-ensemble. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 369–385
43. Liu X, Xiao T, Si S, Cao Q, Kumar S, Hsieh C-J (2020) How does noise help robustness? explanation and exploration under the neural sde framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 282–290
44. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
45. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
46. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto
47. Carlini N, Wagner D (2017) Adversarial examples are not easily detected: Bypassing ten detection methods. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp 3–14
48. Ma X, Li B, Wang Y, Erfani SM, Wijewickrema SNR, Schoenebeck G, Song D, Houle ME, Bailey J (2018) Characterizing adversarial subspaces using local intrinsic dimensionality. In: 6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings. OpenReview.net, Vancouver
49. Xie C, Tan M, Gong B, Yuille AL, Le QV (2020) Smooth adversarial training. CoRR arXiv:2006.14536
50. Tramèr F, Kurakin A, Papernot N, Goodfellow IJ, Boneh D, McDaniel PD (2018) Ensemble adversarial training: Attacks and defenses. In: 6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings. OpenReview.net, Vancouver
51. Xie C, Wu Y, van der Maaten L, Yuille AL, He K (2019) Feature denoising for improving adversarial robustness. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 501–509
52. Borkar T, Heide F, Karam L (2020) Defending against universal attacks through selective feature regeneration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 709–719
53. Liao F, Liang M, Dong Y, Pang T, Hu X, Zhu J (2018) Defense against adversarial attacks using high-level representation guided denoiser. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1778–1787
54. Mustafa A, Khan SH, Hayat M, Shen J, Shao L (2019) Image super-resolution as a defense against adversarial attacks. IEEE Trans Image Process 29:1711–1724
55. Prakash A, Moran N, Garber S, DiLillo A, Storer J (2018) Deflecting adversarial attacks with pixel deflection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8571–8580
56. Papernot N, McDaniel P, Wu X, Jha S, Swami A (2016) Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE Symposium on Security and Privacy (SP). IEEE, pp 582–597
57. Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A (2017) Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security, pp 506–519

58. Gu S, Rigazio L (2015) Towards deep neural network architectures robust to adversarial examples. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings, San Diego
59. Nayebi A, Ganguli S (2017) Biologically inspired protection of deep networks from adversarial attacks. CoRR arXiv:1703.09202
60. Nguyen A, Yosinski J, Clune J (2015) Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 427–436
61. Grosse K, Manoharan P, Papernot N, Backes M, McDaniel PD (2017) On the (statistical) detection of adversarial examples. CoRR arXiv:1702.06280
62. Metzen JH, Genewein T, Fischer V, Bischoff B (2017) On detecting adversarial perturbations. In: 5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings. OpenReview.net, Toulon
63. Wang S, Gong Y (2021) Adversarial example detection based on saliency map features. Appl Intell:1–14
64. Eniser HF, Christakis M, Wüstholz V (2020) RAID: randomized adversarial-input detection for neural networks. CoRR arXiv:2002.02776
65. Meng D, Chen H (2017) Magnet: a two-pronged defense against adversarial examples. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp 135–147
66. Potra FA, Wright SJ (2000) Interior-point methods. J Comput Appl Math 124(1-2):281–302
67. Bendale A, Boult TE (2016) Towards open set deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1563–1572
68. Ruder S (2017) An overview of multi-task learning in deep neural networks. CoRR arXiv:1706.05098
69. Vandenhende S, Georgoulis S, Proesmans M, Dai D, Gool LV (2020) Revisiting multi-task learning in the deep learning era. CoRR arXiv:2004.13379
70. Kendall A, Gal Y, Cipolla R (2018) Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7482–7491
71. Chen Z, Badrinarayanan V, Lee C-Y, Rabinovich A (2018) Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In: Dy JG, Krause A (eds) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Proceedings of Machine Learning Research, vol 80. PMLR, Stockholmsmässan, pp 793–802
72. Guo M, Haque A, Huang D-A, Yeung S, Fei-Fei L (2018) Dynamic task prioritization for multitask learning. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 270–287
73. Sener O, Koltun V (2018) Multi-task learning as multi-objective optimization. In: Advances in Neural Information Processing Systems, pp 527–538
74. Zhang L, Tan Z, Song J, Chen J, Bao C, Ma K (2019) Scan: A scalable neural networks framework towards compact and efficient models. In: Advances in Neural Information Processing Systems, pp 4027–4036
75. Zhang L, Yu M, Chen T, Shi Z, Bao C, Ma K (2020) Auxiliary training: Towards accurate and robust models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 372–381
76. Zhang L, Song J, Gao A, Chen J, Bao C, Ma K (2019) Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In: Proceedings of the IEEE International Conference on Computer Vision, pp 3713–3722
77. Biggio B, Corona I, Maiorca D, Nelson B, Šrndić N, Laskov P, Giacinto G, Roli F (2013) Evasion attacks against machine learning at test time. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 387–402
78. Andriushchenko M, Croce F, Flammarion N, Hein M (2020) Square attack: a query-efficient black-box adversarial attack via random search. In: European Conference on Computer Vision. Springer, pp 484–501
79. Chen J, Jordan MI, Wainwright MJ (2020) Hopskipjumpattack: A query-efficient decision-based attack. In: 2020 ieee symposium on security and privacy (sp). IEEE, pp 1277–1294
80. Storn R, Price KV (1997) Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
81. Pang T, Du C, Dong Y, Zhu J (2018) Towards robust detection of adversarial examples. In: Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, pp 4584–4594
82. Aldahdooh A, Hamidouche W, Fezza SA, Déforges O (2022) Adversarial example detection for dnn models: A review and experimental comparison. Artif Intell Rev

Ahmed Aldahdooh received his Master's degree in Multimedia and Data Management from Polytech Nantes, Nantes University, France in 2014. Then he joined LS2N, Polytech Nantes, to obtain the PhD degree in IT and its applications in 2017. In 2020, he joined IETR at INSA Rennes as a research engineer. His main research interests are, content-aware video delivery, video quality, image and video processing, error concealment, deep learning, safety of deep learning, and adversarial examples detection.

Wassim Hamidouche received Master's and Ph.D. degrees both in Image Processing from the University of Poitiers (France) in 2007 and 2010, respectively. From 2011 to 2013, he was a junior scientist in the video coding team of Canon Research Center in Rennes (France). He was a post-doctoral researcher from Apr. 2013 to Aug. 2015 with VAADER team of IETR where he worked under collaborative project on HEVC video standardisation. Since Sept. 2015 he is an Associate Professor at INSA Rennes and a member of the VAADER team of IETR Lab. He has joined the Advanced Media Content Lab of b<>com IRT Research Institute as an academic member in Sept. 2017. His research interests focus on video coding and multimedia security. He is the author/coauthor of more than one hundred and thirty (+130) papers at top journals and conferences in Image Processing, two MPEG standards, two patents, several MPEG contributions, public datasets and open source software projects.

Olivier Déforges received the Ph.D. degree in image processing in 1995. He is a Professor with the National Institute of Applied Sciences (INSA) of Rennes. In 1996, he joined the Department of Electronic Engineering, INSA of Rennes, Scientic and Technical University. He is a member of the Institute of Electronics and Telecommunications of Rennes (IETR), UMR CNRS 6164 and leads the IMAGE Team, IETR Laboratory including 40 researchers. He has authored over 130 technical papers. His principal research interests are image and video lossy and lossless compression, image understanding, fast prototyping, and parallel architectures. He has also been involved in the ISO/MPEG standardization group since 2007.